



Spectral Learning of Sequential Systems

by

Michael Thon

A thesis submitted in partial fulfillment
of the requirements for the degree of

**Doctor of Philosophy
in Computer Science**

Approved Dissertation Committee:

Dr. Herbert Jaeger

Jacobs University Bremen, Germany

Richard S. Sutton, PhD

University of Alberta, Canada

Dr. Mathias Bode

Jacobs University Bremen, Germany

Date of Defense: 1 December 2017

Department of Computer Science & Electrical Engineering

Statutory Declaration

(Declaration on Authorship of a Dissertation)

I, Michael Thon, hereby declare, under penalty of perjury:

I have written this PhD thesis independently, unless where clearly stated otherwise. I have used only the sources, the data and the support that I have clearly mentioned.

This PhD thesis has not been submitted for the conferral of a degree elsewhere.

I am aware of the consequences of a deliberately or negligently wrongly submitted affidavit, in particular the punitive provisions of §156 and §161 of the Criminal Code (up to 1 year imprisonment or a fine at delivering a negligent or 3 years or a fine at a knowingly false affidavit).

Bremen, August 28, 2017

Signature

Abstract

Stochastic multiplicity automata (SMA) are weighted nondeterministic automata that generalize probabilistic automata and have been used in the context of probabilistic grammatical inference. Observable operator models (OOMs) are a generalization of hidden Markov models, which in turn are models for discrete-valued stochastic processes used ubiquitously in the context of speech recognition and bio-sequence modeling. Predictive state representations (PSRs) extend OOMs to stochastic input-output systems and are employed in the context of agent modeling and planning.

In this thesis we first unify these statistical models under the framework of sequential systems (SSs), which are abstract linear algebraic models for certain types of functions on words. The required parts of the theory of SS are presented in a self-contained and easily accessible fashion. We proceed to show how many of the available learning algorithms for these models can be understood as instances of a common learning procedure.

Our focus lies on the state-of-the-art spectral learning algorithm. We generalize two recent learning algorithms for OOMs that are based on entirely different learning principles and show that these are in fact equivalent or closely related to spectral learning. We improve methods for two key steps that are required in all learning algorithms, namely the selection of “characteristic” words and of a suitable model dimension. Furthermore, we develop a new weighted spectral learning algorithm that incorporates weights that take the precision of individual estimates into account and show empirically that these modifications indeed improve standard spectral learning.

Finally, we address the previously unresolved problem of learning OOMs from data that contains missing values.

To my family and friends.

Acknowledgements

First, I would like to thank my PhD advisor, Dr. Herbert Jaeger, for his guidance and continued support. I also thank Richard Sutton and Dr. Mathias Bode for being on my thesis committee, and my colleagues over the years for many fruitful scientific discussions.

I gratefully acknowledge the funding received by the German Research Foundation (DFG) for preliminary research under the project JA 1210/5-1.

My special thanks for proof-reading of the thesis go to my mom, Elizabeth, who read the whole manuscript even though she didn't understand much of the mathematics, and to Hao Wu for his last-minute comments.

Finally, and foremost, I would like to thank those closest to me, especially my wife Friederike, but also our kids Hauke and Nika, and my parents for their love, encouragement and enduring me in times of frustration. Without their support I would not have been able to complete this thesis.

Contents

1	Introduction	1
2	Notation	5
3	Sequential Systems	7
4	Versions of Sequential Systems	17
4.1	Multiplicity Automata and Weighted Automata	18
4.1.1	Stochastic Multiplicity Automata and Stochastic Languages . . .	20
4.2	Observable Operator Models and Stochastic Processes	21
4.2.1	Relation to Hidden Markov Models	22
4.2.2	Relation to Stochastic Multiplicity Automata	23
4.2.3	Historical Remarks	25
4.3	Predictive State Representations and Controlled Processes	26
4.3.1	Relation to Partially Observable Markov Decision Processes . . .	28
4.3.2	IO-OOMs, Interpretable IO-OOMs, PSRs, and TPSRs	28
4.3.3	Historical Remarks	29
4.4	Extensions	30
5	Learning Sequential Systems	33
5.1	Obtaining Estimates $\hat{f}(\bar{x})$	36
5.1.1	Variance Estimates	39
5.2	Choosing Indicative and Characteristic Words	43
5.2.1	A General Selection Method	45
5.3	Determining the Model Dimension	49
5.3.1	Numerical Rank	50
5.4	Selecting the Characterizer and Indicator	53
5.4.1	Selection or Grouping of Rows and Columns of $\hat{F}$	54

CONTENTS

5.4.2	Spectral Methods	54
5.4.3	A Weighted Spectral Learning Algorithm	57
5.4.3.1	Weighted Regression and Low-Rank Approximation . .	58
5.4.3.2	How to Make Use of Weights	65
5.4.4	A Row and Column Weighted Spectral Algorithm	68
5.4.5	The Error Controlling Algorithm	72
5.4.6	The Efficiency Sharpening Algorithm	78
5.5	Computational Complexity	83
5.6	Normalization, Stabilization, Sampling	85
6	Empirical Investigations	89
6.1	Benchmark Problems and Datasets	90
6.1.1	Benchmarks for Controlled Processes	91
6.1.2	Benchmarks for Stochastic Processes	92
6.1.2.1	Real-world Data	94
6.1.2.2	Synthetic Data	95
6.1.3	Default Evaluation on Benchmarks	98
6.2	Selection of Indicative and Characteristic Words	101
6.3	Selecting the Target Dimension	109
6.4	Comparison of Weighted Spectral Learning Variants	118
7	Learning with Missing Values	123
7.1	Current Restrictions	127
7.2	Modified Estimates $\hat{f}(\bar{y})$	128
7.3	Empirical Investigation	129
7.3.1	Results	132
8	Conclusion	139
	References	143

1

Introduction

Multiplicity automata (MA) [Schützenberger, 1961] are weighted nondeterministic automata that generalize both finite and probabilistic automata. The discovery that MA are efficiently learnable [Bergadano and Varricchio, 1994; Ohnishi et al., 1994] in the exact learning model of Angluin [Angluin, 1987] sparked an interest in these, and several versions have been studied. One such version are stochastic multiplicity automata (SMA) which model rational stochastic languages and have been used in the context of probabilistic grammatical inference [Denis et al., 2006; Bailly et al., 2009]. Independent of this line of research, hidden Markov models (HMMs) [see Rabiner, 1989] for discrete-valued stochastic processes have been extensively studied and are now a standard tool in many pattern recognition domains such as speech recognition, natural language processing, and bio-sequence modeling. Observable operator models (OOMs) are a generalization of HMMs that were introduced by Jaeger [1998] following previous work on deciding the equivalence of HMMs [Ito et al., 1992] and have recently found applications in the field of molecular dynamics [Wu et al., 2015; Nüske et al., 2017]. Finally, predictive state representations (PSRs) are models for stochastic input-output systems developed by Littman, Sutton, and Singh [2001] and inspired by OOMs. Such PSRs generalize partially observable Markov decision processes (POMDPs) [Kaelbling et al., 1998] and have been used in the context of agent modeling and planning [James et al., 2004; James and Singh, 2005; Wolfe and Singh, 2006; Boots et al., 2010]. As it turns out, all of these models are closely related, though this is not widely perceived, due in part to the disjoint scientific communities.

SMA, OOMs, and PSRs each model some form of probability distribution. A central task common to all cases is, therefore, to estimate a model from given sample data. This may also be referred to as learning, system identification, or model induc-

1. INTRODUCTION

tion depending on the context. Learning algorithms with state-of-the-art performance are the efficiency sharpening (ES) algorithm [Jaeger et al., 2006b] and the error controlling (EC) algorithm [Zhao et al., 2009a,b] for OOMs, as well as a spectral learning algorithm that was discovered independently for the case of PSRs [Rosencrantz et al., 2004], SMA [Bailly et al., 2009], and HMMs via OOMs [Hsu et al., 2009]. Spectral algorithms find a low-dimensional representation of input data by computing a low-rank matrix approximation to a certain data affinity matrix and are a generic tool in statistics and machine learning. Spectral algorithms have applications far beyond these models and are, for instance, widely used for spectral clustering in the context of image segmentation [Shi and Malik, 2000].

The first contribution of this thesis is the unification of SMA, OOMs, and PSRs under the framework of sequential systems (SSs), which are a purely algebraic characterization of MA that goes back to Carlyle and Paz [1971]. In Chapter 3 we cover the required parts of the theory of SSs in a self-contained and easily accessible fashion, while in Chapter 4 we present SMA, OOMs, and PSRs as instances of SSs with specific additional constraints and explore the precise relationships between these models. There has been much confusion over the precise relationship between PSRs and OOMs, which we address in Section 4.3.2. In Chapter 5 we present a general Algorithm 4 to learn such models from data and show that many of the known learning algorithms can be understood as variations of this basic learning approach. In particular, we present the spectral learning algorithm in this framework as a general learning algorithm for SMA, OOMs, and PSRs in Section 5.4.2. In Sections 5.4.5 and 5.4.6 we give simplified yet generalized accounts of the EC algorithm as well as the ES algorithm and show that these are equivalent or closely related to the spectral learning algorithm, even though these algorithms are derived from very different learning principles, which leads to a much better understanding of the workings of these algorithms. These parts of the thesis have already been published by Thon and Jaeger [2015] and largely coincide with that publication. We should point out that these contributions are entirely due to the first author Michael Thon, while Herbert Jaeger provided valuable guidance and assisted in the preparation of the manuscript.

The second contribution of this thesis consists of improvements to the learning algorithms. The general learning Algorithm 4 makes the steps involved in learning a SS model from data explicit, which in turn allows us to study each of these steps separately in more detail. In Section 5.2.1 we present a novel and more general method of selecting required sets of indicative and characteristic words, and in Section 5.3.1 we refine the previously proposed methods of selecting a suitable model dimension

based on the numerical rank of the estimated system matrix. Our main contribution, however, is the introduction of weights to the spectral learning algorithm that measure the precision of relevant estimates. We explain how to accomplish this in Section 5.4.3, where we present the novel weighted spectral learning Algorithm 8, and then derive a simplified row and column weighted spectral learning Algorithm 9 in the following Section 5.4.4. In Section 5.1 we explain how to obtain the required precision estimates in a way that is suitable for use in the weighted spectral learning algorithms.

The third contribution of this thesis is the comprehensive empirical investigation that we conduct in Chapter 6. We carefully select benchmark problems to cover interesting processes of higher and lower dimension as well as with large and small alphabet sizes. We then use these benchmark problems to systematically investigate the impact of various algorithmic choices for each of the steps in the general learning Algorithm 4. This has not been done previously and we gain many new and helpful insights from this investigation, which lead to informed recommendations for various algorithmic design decisions. We also compare our novel weighted spectral learning algorithms to the standard spectral learning algorithm, which we use as a baseline.

The fourth and final contribution of this thesis is Chapter 7, where we devise a learning algorithm for OOMs that can deal with training data that contains missing values. This is based on viewing the missingness as inputs and learning a model for the resulting input-output process instead. We specify a set of conditions under which this approach is guaranteed to work, present improved estimators tailored to this task, and investigate our approach empirically on the same benchmarks that we used in the previous Chapter 6.

All of the empirical results contained in this thesis are obtained using the open source “toolkit for observable operating modeling” by Thon [2017], which is a fast implementation of the core routines and learning algorithms for (IO)-OOMs written in C++ and Python and developed alongside the thesis research. The scripts used to compute the results and graphs contained in this thesis are provided with the toolkit, so that the computations can easily be reproduced exactly.

1. INTRODUCTION

2

Notation

Let Σ be a finite alphabet. We will denote the set of words over this alphabet by Σ^* , which includes the *empty word* ε , while Σ^l and $\Sigma^{\leq l}$ denote the sets of words of length l or of length less than or equal to l , respectively. Words in Σ^* will be denoted by variables with a bar over them, e.g. $\bar{x}, \bar{y}, \bar{z}, \bar{s} \in \Sigma^*$. If we wish to make it explicit that we mean only a word of length one, i.e., a symbol from the alphabet Σ , we will use normal variables, as in $x, y, z \in \Sigma$. For any words $\bar{x}, \bar{y} \in \Sigma^*$, let $\bar{x}\bar{y}$ be the concatenation of words and $|\bar{x}|$ denote the length of the word $\bar{x}$. We will also allow concatenation of words and sets of words. For instance, if $Z \subset \Sigma^*$ is a set of words, then we will use $\bar{x}Z\bar{y}$ to denote the set of words of the form $\bar{x}\bar{z}\bar{y}$ for any $\bar{z} \in Z$, i.e., $\bar{x}Z\bar{y} = \{\bar{x}\bar{z}\bar{y} | \bar{z} \in Z\}$. We will use $\bar{x} = x_1 \dots x_n \in \Sigma^*$ if we wish to refer to the individual symbols x_k of a word $\bar{x}$. Sometimes the symbols will in fact be pairs of input and output symbols. In this case, we will write the alphabet Σ as the Cartesian product of an *input alphabet* Σ_I and an *output alphabet* Σ_O , i.e., $\Sigma = \Sigma_I \times \Sigma_O$. We will refer to input symbols by $a \in \Sigma_I$ (for “action”), and output symbols by $o \in \Sigma_O$ (for “observation”) so that we can omit the tuple notation and write, for instance $\bar{x} = a_1 o_1 \dots a_n o_n \in \Sigma^*$.

To define matrices, and also to index elements in a matrix, we will use a square bracket notation with lower indices. For this notation, we regard vectors as a matrix consisting of a single column, i.e., of size $n \times 1$, row vectors as a matrix of size $1 \times n$, and scalars as a matrix of size 1×1 . To explain the notation, let $M_1, \dots, M_n$ be matrices (which may be vectors or even just scalars). By $[M_i]_{i \in I} = [M_1, \dots, M_n]$ for $I = \{1, \dots, n\}$ we will denote the matrix obtained by stacking the matrices M_i horizontally, which is only well-defined if the matrices M_i all have the same number of rows. In the case of scalars this will give a row vector and we will therefore write $[M_i]_{i \in I}^\top$ to denote the (column) vector of the scalars M_i , where $\cdot^\top$ denotes matrix

2. NOTATION

transposition. In general, if we wish to stack the M_i vertically and the M_i are not scalars, we will use the slightly awkward notation $[M_i^\top]_{i \in I}^\top$, which requires the M_i to all have the same number of columns. Finally, for scalars $M_{i,j}$ only, we will use the notation $[M_{i,j}]_{i \in I, j \in J}$ to denote the matrix that has entry $M_{i,j}$ in the row indexed by i and column indexed by j . Note that we will be using sets of words $X, Y \subset \Sigma^*$ with some implicit ordering as index sets. For example, for a function $f : \Sigma^* \rightarrow \mathbb{R}$, the matrices $F_{Y,X} = [f(\bar{x}\bar{y})]_{\bar{y} \in Y, \bar{x} \in X}$ and $F_{Z,Y,X} = [f(\bar{x}z\bar{y})]_{\bar{y} \in Y, \bar{x} \in X}$ that have columns indexed by the words $\bar{x} \in X$ and rows indexed by the words $\bar{y} \in Y$ will play a central role. Conversely, if we have already defined a matrix M , we will use the notation $[M]_{i,j}$ to denote the entry in the row indexed by i and column indexed by j , while $[M]_j$ will denote the entire column indexed by j . To refer to the row indexed by i , we will use $[M^\top]_i^\top$.

For matrices $M \in \mathbb{R}^{m \times n}$, $M_1, M_2, \dots, M_k$ and a vector v , we will use $M^\top$ to denote the matrix transpose, M^{-1} to denote the matrix inverse, $M^\dagger$ for the Moore-Penrose pseudo-inverse, $\text{tr}(M)$ for the trace, $M_1 \otimes M_2$ to denote the Kronecker product of M_1 and M_2 , and $\text{vec}(M)$ to denote the column-wise *vectorization* of M , i.e., the vector $[[M]_1^\top, \dots, [M]_n^\top]^\top$. We will use $\mathbf{I}_n$ to denote the identity matrix of size $n \times n$, or simply write $\mathbf{I}$ if the size is clear from context, and use the notation $\mathbb{1}_n = [1, \dots, 1]^\top$ for the n -dimensional vector of ones, or simply write $\mathbb{1}$ if the size is clear from context. We denote by $\mathbf{D}(v)$ the diagonal matrix with the vector v on its diagonal and by $\mathbf{D}(M_1, M_2, \dots, M_k)$ the block-diagonal matrix with the matrices $M_1, M_2, \dots, M_k$ on the diagonal. Finally, for vectors $v_1, \dots, v_k$, let $\text{span}(\{v_1, \dots, v_k\}) = \text{span}([v_1, \dots, v_k])$ denote the linear subspace spanned by these vectors.

For random variables, e.g., $X_t : \Omega \rightarrow \Sigma$, we will use the shorthand $\mathbb{P}(\bar{x})$ or $\mathbb{P}(x_1 \dots x_n)$ to denote the joint probability $\mathbb{P}(X_1 = x_1, \dots, X_n = x_n)$. We use $\mathbb{E}[X]$ and $\mathbb{V}[X]$ to denote the expectation and variance of a random variable X , respectively, if these are defined. In case X is a random vector, $\mathbb{V}[X]$ is the covariance matrix of X . When estimating some parameter p , we will usually use the notation $\hat{p}$ to denote a parameter estimator, which is a random variable. However, we will also use $\hat{p}$ to refer to an estimated value for p . The meaning will be clear from the context.

3

Sequential Systems

In this chapter we first introduce some basic definitions. Then we present the theory of what are called sequential systems (SSs). This can be seen as a generalized treatment of observable operator models (OOMs) that is also applicable to the cases of predictive state representations (PSRs) and multiplicity automata (MA). This theory is not new and in fact goes back to Schützenberger [1961], who initiated the algebraic treatment of MA, to Carlyle and Paz [1971], who coined the term *sequential systems*, and to Fliess [1974]. The same theory has been rediscovered several times in various forms, for instance for OOMs [Jaeger, 2000b] and PSRs [Singh et al., 2004]. However, the original treatment was purely abstract, without the desire to turn any of the results into computational models. Many of the central proofs were non-constructive and involved, since they concerned more general versions of the theory than we require, and were therefore not of much use for our purposes. On the other hand, the presentation of this basic theory in the context of OOMs and PSRs is less general, as the proofs make much use of properties specific to OOMs and PSRs, for instance that one is dealing with stochastic processes. Here, we present the basic general theory in a concise, self-contained fashion with constructive proofs that can readily be turned into algorithms.

Let Σ be a finite alphabet. We will be interested in characterizing functions $f : \Sigma^* \rightarrow K$ for $K = \mathbb{R}$ or $K = \mathbb{C}$, since these can be used to describe probabilistic languages, stochastic processes and controlled processes (see Definitions 20, 22, and 30). These form a K -vector space which we denote by $K\langle\langle\Sigma\rangle\rangle$. For a given function $f : \Sigma^* \rightarrow K$, we define the *system matrix* F as the infinite matrix $F = [f(\bar{x}\bar{y})]_{\bar{y} \in \Sigma^*, \bar{x} \in \Sigma^*}$. Note that this is the transpose of what is sometimes called the *Hankel matrix*. Furthermore, for a given function f we define the functions $f_{\bar{x}} : \Sigma^* \rightarrow K$ by setting $f_{\bar{x}}(\bar{y}) = f(\bar{x}\bar{y})$ for any sequences $\bar{x}, \bar{y} \in \Sigma^*$ and note that these functions correspond to the columns of

3. SEQUENTIAL SYSTEMS

the system matrix F . Let $\mathcal{F} = \text{span}\{f_{\bar{x}} \mid \bar{x} \in \Sigma^*\}$ be the linear space spanned by these functions / the columns of F . Clearly, $\mathcal{F} \subset K\langle\langle\Sigma\rangle\rangle$. We define $\text{rank}(f) = \text{rank}(F) = \text{rank}(\mathcal{F})$.

Definition 1. A d -dimensional sequential system (SS) over the alphabet Σ is a structure $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$, which consists of an initial state vector $\omega_\varepsilon \in K^d$, a matrix $\tau_z \in K^{d \times d}$ for each $z \in \Sigma$, and an evaluation function $\sigma : K^d \rightarrow K$. For $\bar{x} = x_1 \dots x_n \in \Sigma^*$ let $\tau_{\bar{x}} = \tau_{x_n} \dots \tau_{x_1}$, and let $\omega_{\bar{x}} = \tau_{\bar{x}} \omega_\varepsilon$, which we call a state of the SS $\mathcal{M}$. Let $\tau_\Sigma = \sum_{z \in \Sigma} \tau_z$. If the function σ is linear, the SS $\mathcal{M}$ is called linear.

The reason we are interested in such SS structures is that they can be used to describe functions $f : \Sigma^* \rightarrow K$ in the following way.

Definition 2. For a given SS $\mathcal{M}$, we define its (external) function to be

$$f_{\mathcal{M}} : \Sigma^* \rightarrow K, \quad f_{\mathcal{M}}(\bar{x}) = \sigma \tau_{\bar{x}} \omega_\varepsilon \quad (1)$$

Furthermore, we define the rank of the SS $\mathcal{M}$ to be $\text{rank}(\mathcal{M}) = \text{rank}(f_{\mathcal{M}})$.

Here, we will be dealing only with the linear case, so σ will be a linear functional that is represented by a row vector, i.e., $\sigma^\top \in K^d$.

We begin with a technical result that lies at the heart of the whole theory.

Proposition 3. Let $f : \Sigma^* \rightarrow K$ be given. If $\text{rank}(f) = d < \infty$, then there exist linear operators $\tilde{\tau}_z : \mathcal{F} \rightarrow \mathcal{F}$ for each $z \in \Sigma$ and a linear functional $\tilde{\sigma} : \mathcal{F} \rightarrow K$ that satisfy $\tilde{\tau}_z(f_{\bar{x}}) = f_{\bar{x}z}$ and $\tilde{\sigma}(f_{\bar{x}}) = f(\bar{x})$ for all $\bar{x} \in \Sigma^*$. Furthermore, $\tilde{\sigma}(\tilde{\tau}_{\bar{x}}(f_\varepsilon)) = f(\bar{x})$ for all $\bar{x} \in \Sigma^*$, where $\tilde{\tau}_{\bar{x}} = \tilde{\tau}_{x_n} \circ \dots \circ \tilde{\tau}_{x_1}$.

Proof. Let $X \subset \Sigma^*$ be an index set denoting a maximal set of linearly independent columns of the matrix F . Then clearly, $\mathcal{B} = \{f_{\bar{x}} \mid \bar{x} \in X\}$ is a basis for $\mathcal{F}$. Define linear operators $\tilde{\tau}_z$ and a linear functional $\tilde{\sigma}$ by their action on these basis elements:

- $\tilde{\tau}_z(f_{\bar{x}}) = f_{\bar{x}z}$ for all $\bar{x} \in X$ and $z \in \Sigma$,
- $\tilde{\sigma}(f_{\bar{x}}) = f_{\bar{x}}(\varepsilon) = f(\bar{x})$ for all $\bar{x} \in X$.

We will show that $\tilde{\tau}_z(f_{\bar{y}}) = f_{\bar{y}z}$ and $\tilde{\sigma}(f_{\bar{y}}) = f(\bar{y})$ in fact holds for all $\bar{y} \in \Sigma^*$. For this, let $\bar{y} \in \Sigma^*$. Then $f_{\bar{y}} = \sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}}$ for suitable coordinates $\lambda_{\bar{x}}$, and $f_{\bar{y}z} = \sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}z}$, since for any $\bar{z} \in \Sigma^*$, we have $f_{\bar{y}z}(\bar{z}) = f_{\bar{y}}(z\bar{z}) = \sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}}(z\bar{z}) = \sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}z}(\bar{z})$. Therefore, $\tilde{\tau}_z(f_{\bar{y}}) = \tilde{\tau}_z(\sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}}) = \sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}z} = f_{\bar{y}z}$, and $\tilde{\sigma}(f_{\bar{y}}) = \tilde{\sigma}(\sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}}) = \sum_{\bar{x} \in X} \lambda_{\bar{x}} f_{\bar{x}}(\varepsilon) = f_{\bar{y}}(\varepsilon) = f(\bar{y})$.

Finally, $\tilde{\sigma}(\tilde{\tau}_{\bar{y}}(f_\varepsilon)) = \tilde{\sigma}(f_{\bar{y}}) = f(\bar{y})$ for all $\bar{y} \in \Sigma^*$. ■

The above proposition establishes a crucial property that makes this theory appealing, as it means that the functions $f = f_\varepsilon$, $f_{\bar{x}} = \tilde{\tau}_{\bar{x}}(f)$, the linear operators $\tilde{\tau}_z$, and the linear functional $\tilde{\sigma}$ have coordinate representations as vectors and matrices with respect to some basis $\mathcal{B}$ for $\mathcal{F}$. Note that this remains true even if $\text{rank}(f) = \infty$, but the coordinate representations will then be infinite and of little practical use. The property $f(\bar{x}) = \tilde{\sigma}(\tilde{\tau}_{\bar{x}}(f_\varepsilon))$ (cf. Equation (1)) means that the function f is fully described by the data $(\tilde{\sigma}, \{\tilde{\tau}_z\}, f_\varepsilon)$. If these are given in some coordinate representation, then we have a SS representation.

Proposition 4. *Let $f : \Sigma^* \rightarrow K$ be given. If $\text{rank}(f) = d < \infty$, then there exists a d -dimensional SS $\mathcal{M}$ such that $f = f_{\mathcal{M}}$.*

Proof. Let $\mathcal{B}$ be a basis for $\mathcal{F}$, and let $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be the coordinate representations of $(\tilde{\sigma}, \{\tilde{\tau}_z\}, f_\varepsilon)$ with respect to $\mathcal{B}$, where we are using the definitions for $\tilde{\sigma}$ and $\tilde{\tau}_z$ from the above Proposition 3. Then for any $\bar{x} \in \Sigma^*$, we have $f(\bar{x}) = \tilde{\sigma}(\tilde{\tau}_{\bar{x}}(f_\varepsilon)) = \sigma \tau_{\bar{x}} \omega_\varepsilon = f_{\mathcal{M}}(\bar{x})$. ■

Note that for the SS $\mathcal{M}$ constructed in Proposition 4 as a coordinate representation with respect to some basis $\mathcal{B}$ of $\mathcal{F}$, the states $\omega_{\bar{x}} = \tau_{\bar{x}} \omega_\varepsilon$ will be the coordinate representations of the functions $f_{\bar{x}} = \tilde{\tau}_{\bar{x}}(f)$ with respect to the basis $\mathcal{B}$. Also note that due to Equation (1) we may evaluate $f(\bar{x})$ using the SS $\mathcal{M}$ without knowing the basis $\mathcal{B}$.

The above proposition suggests that two SSs might describe the same function f if and only if they are representations for f with respect to different bases for $\mathcal{F}$. However, this is only correct for so-called *minimal* SS, as will be detailed in the following.

Definition 5. *Two SSs $\mathcal{M}$ and $\mathcal{M}'$ are equivalent, denoted by $\mathcal{M} \cong \mathcal{M}'$, if they define the same function, i.e., if $f_{\mathcal{M}} = f_{\mathcal{M}'}$. It is clear that this notion is an equivalence relation on the set of all SSs.*

We now introduce concepts needed to characterize the equivalence on SSs. We give such a characterization for *minimal* SSs in Proposition 14. For this, we introduce the concept of minimal SSs and give criteria for minimality in Corollary 10 as well as a procedure in Algorithm 2 to construct an equivalent minimal SS.

Definition 6. *For a given SS $\mathcal{M}$ we call the linear spaces $W = \text{span}\{\tau_{\bar{x}} \omega_\varepsilon \mid \bar{x} \in \Sigma^*\}$ the state space and $\tilde{W} = \text{span}\{(\sigma \tau_{\bar{x}})^\top \mid \bar{x} \in \Sigma^*\}$ the co-state space of $\mathcal{M}$.*

Definition 7. *We call a d -dimensional SS $\mathcal{M}$ trimmed if it has full state and co-state spaces, i.e., if $W = \tilde{W} = K^d$. We call a SS minimal if no equivalent model of lower dimension exists.*

3. SEQUENTIAL SYSTEMS

It will turn out in Corollary 10 that a SS is minimal if and only if it is trimmed. But first, we show how to construct bases for the state (and co-state) space of a SS.

Proposition 8. *The following Algorithm 1 constructs a basis $\mathcal{B}$ for the state space W of a given d -dimensional SS in time $\mathcal{O}((d + |\Sigma|)d^3)$ (the construction of a basis $\tilde{\mathcal{B}}$ for the co-state space $\tilde{W}$ is analogous).*

Algorithm 1: Compute a basis $\mathcal{B}$ for the state space W of a given SS

```

 $\mathcal{B} \leftarrow \{\}, \mathcal{C} \leftarrow \{\omega_\varepsilon\}$ 
while  $|\mathcal{C}| > 0$  do
     $\omega \leftarrow \text{some element of } \mathcal{C}, \mathcal{C} \leftarrow \mathcal{C} \setminus \{\omega\}$ 
    if  $\omega$  is linearly independent of  $\mathcal{B}$  then
         $\mathcal{B} \leftarrow \mathcal{B} \cup \{\omega\}$ 
         $\mathcal{C} \leftarrow \mathcal{C} \cup \{\tau_z \omega \mid z \in \Sigma\}$ 

```

Proof. At any time during the run of the algorithm, $\mathcal{B}$ is a set of linearly independent vectors. Furthermore, the set $\mathcal{C}$ of “candidate vectors” increases by $|\Sigma|$ elements each time a new vector is added to the set $\mathcal{B}$ but decreases by one element each run through the main loop. Therefore, the algorithm terminates after at most $d|\Sigma| + 1$ runs through the main loop, since there are at most d linearly independent vectors that can be added to $\mathcal{B}$. Next we examine the runtime of the algorithm. Checking ω for linear independence from $\mathcal{B}$ can be done by checking $P_{\mathcal{B}}\omega = \omega$ in time $\mathcal{O}(d^2)$ if the orthogonal projection matrix $P_{\mathcal{B}}$ onto $\text{span}(\mathcal{B})$ is known. This check is performed at most $d|\Sigma| + 1$ times, yielding a complexity of $\mathcal{O}(d^3|\Sigma|)$. Clearly, the matrix $P_{\mathcal{B}}$ must be updated every time a vector is added to $\mathcal{B}$, which is an $\mathcal{O}(d^3)$ operation that needs to be performed at most d times, giving a total complexity of $\mathcal{O}(d^4)$. Finally, every time a vector ω is added to $\mathcal{B}$, the set $\mathcal{C}$ is increased by $|\Sigma|$ vectors, each of which requires time $\mathcal{O}(d^2)$ to be computed from ω , for a total time complexity of $\mathcal{O}(d^3|\Sigma|)$. Adding these together gives the claimed time complexity.

Finally, we show that the returned set $\mathcal{B}$ is indeed a basis of the state space W . Observe that for all $\omega \in \mathcal{B}$ and for all $z \in \Sigma$, the vectors $\tau_z \omega$ have been added as “candidate vectors” to the set $\mathcal{C}$ at some point during the run of the algorithm – namely when ω was added to $\mathcal{B}$. Each of these vectors is checked in turn and is at that point either linearly dependent on $\mathcal{B}$, or added to $\mathcal{B}$. Therefore, these vectors $\tau_z \omega$ are all linearly dependent on the final set $\mathcal{B}$, i.e., $\tau_z(\mathcal{B}) \subseteq \text{span}(\mathcal{B})$ for all $z \in \Sigma$. By linearity of τ_z this implies that also $\tau_z(\text{span}(\mathcal{B})) \subseteq \text{span}(\mathcal{B})$ for all $z \in \Sigma$. So $\text{span}(\mathcal{B})$ contains ω_ε and is closed under the action of τ_z for all $z \in \Sigma$, which implies that

$\{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in \Sigma^*\} \subseteq \mathbf{span}(\mathcal{B})$. But $\mathcal{B} \subset \{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in \Sigma^*\}$ by construction of $\mathcal{B}$. Together, this implies $\mathbf{span}(\mathcal{B}) = \mathbf{span}(\{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in \Sigma^*\}) = W$. ■

In practice, we will usually select the element w from the set $\mathcal{C}$ of candidate vectors that is most linearly independent of $\mathcal{B}$, for instance maximizing $\|P_{\mathcal{B}}w - w\|$, and stop if the remaining candidate vectors are not sufficiently linearly independent of $\mathcal{B}$, i.e., if $\|P_{\mathcal{B}}w - w\| < \varepsilon$ for some threshold ε . This requires sorting the set of candidate vectors each time a new vector is added to $\mathcal{B}$, which requires $\mathcal{O}(d^2|\Sigma| \log d|\Sigma|)$ additional time.

Note that these bases are by construction of the form $\mathcal{B} = \{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in X\}$ and $\tilde{\mathcal{B}} = \{(\sigma\tau_{\bar{y}})^\top \mid \bar{y} \in Y\}$ for suitable index sets X, Y consisting of words of length at most d , where d is the dimension of the SS. Also, the above procedure allows us to check whether a given SS is trimmed.

The following proposition is the core technical result needed to establish the connection between a SS being trimmed, having full rank, and being minimal.

Proposition 9. *For a d -dimensional SS $\mathcal{M}$, let $\{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in X\}$ and $\{(\sigma\tau_{\bar{y}})^\top \mid \bar{y} \in Y\}$ be bases for W and $\tilde{W}$ respectively, and define $F_{Y,X} = [f_{\mathcal{M}}(\bar{x}\bar{y})]_{\bar{y} \in Y, \bar{x} \in X}$. Then $\text{rank}(\mathcal{M}) = \text{rank}(F_{Y,X}) \leq \min\{|X|, |Y|\} \leq d$. Furthermore, if $|X| = d$ or $|Y| = d$ then $\text{rank}(\mathcal{M}) = \min\{|X|, |Y|\}$.*

Proof. Define $\Pi = [(\sigma\tau_{\bar{x}})^\top]_{\bar{x} \in \Sigma^*}^\top$ and $\Phi = [\tau_{\bar{x}}\omega_\varepsilon]_{\bar{x} \in \Sigma^*}$, as well as $\Pi_Y = [(\sigma\tau_{\bar{y}})^\top]_{\bar{y} \in Y}^\top \in K^{|Y| \times d}$ and $\Phi_X = [\tau_{\bar{x}}\omega_\varepsilon]_{\bar{x} \in X} \in K^{d \times |X|}$. The rows of Π_Y form a basis for the row space of Π and the columns of Φ_X form a basis for the column space of Φ . Now $F = \Pi\Phi$ and $F_{Y,X} = \Pi_Y\Phi_X$. Therefore $\text{rank}(\mathcal{M}) = \text{rank}(F) = \text{rank}(\Pi\Phi) = \text{rank}(\Pi_Y\Phi) = \text{rank}(\Pi_Y\Phi_X) = \text{rank}(F_{Y,X})$. Moreover, $\text{rank}(\Pi_Y) = |Y|$ and $\text{rank}(\Phi_X) = |X|$ imply that $\text{rank}(\Pi_Y\Phi_X) \leq \min\{|X|, |Y|\} \leq d$ as well as $\text{rank}(\Pi_Y\Phi_X) = |X|$ if $|Y| = d$ and $\text{rank}(\Pi_Y\Phi_X) = |Y|$ if $|X| = d$. ■

As a corollary, we obtain the following result, which allows us to check a d -dimensional SS for minimality by checking whether the SS is trimmed, i.e., by constructing bases for the state and co-state space via Algorithm 1 and checking if these have dimension d .

Corollary 10. *Let $\mathcal{M}$ be a d -dimensional SS. The following are equivalent:*

- (i) $\mathcal{M}$ is trimmed
- (ii) $\text{rank}(\mathcal{M}) = d$
- (iii) $\mathcal{M}$ is minimal

3. SEQUENTIAL SYSTEMS

Proof. If $\mathcal{M}$ has full rank, i.e., $\text{rank}(\mathcal{M}) = d$, then $\mathcal{M}$ must be minimal, as any lower-dimensional SS must have a lower rank and therefore cannot be equivalent. Conversely, if $\mathcal{M}$ is minimal, then we must have $\text{rank}(\mathcal{M}) = d$, since by Proposition 4 there exists a $\text{rank}(\mathcal{M})$ -dimensional equivalent SS. By Proposition 9 – and using the notation from the proposition – we see that $\text{rank}(\mathcal{M}) = d \Leftrightarrow |X| = |Y| = d$, i.e., if and only if $\mathcal{M}$ is trimmed. ■

Next, we define the transformation of a SS by linear maps ρ and ρ' . Such transformations will serve as the basic operation on SSs for all conversion operations.

Definition 11. For a d -dimensional SS $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ and any matrices $\rho \in K^{n \times d}$ and $\rho' \in K^{d \times n}$, we define the n -dimensional SS $\rho\mathcal{M}\rho' = (\sigma\rho', \{\rho\tau_z\rho'\}, \rho\omega_\varepsilon)$.

If ρ is non-singular, and $\rho' = \rho^{-1}$, then this transformation will yield an equivalent *conjugated* SS. If the SS is minimal, then this corresponds to a change of basis for the underlying function space $\mathcal{F}$.

Lemma 12. Let $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be a d -dimensional SS, and $\rho \in \mathbb{R}^{d \times d}$ be non-singular. Then $\mathcal{M} \cong \rho\mathcal{M}\rho^{-1}$. We will call $\rho\mathcal{M}\rho^{-1}$ a conjugate of $\mathcal{M}$.

Proof. $\forall \bar{x} \in \Sigma^* : f_{\rho\mathcal{M}\rho^{-1}}(\bar{x}) = (\sigma\rho^{-1})(\rho\tau_{x_n}\rho^{-1}) \cdots (\rho\tau_{x_1}\rho^{-1})(\rho\omega_\varepsilon) = \sigma\tau_{\bar{x}}\omega_\varepsilon = f_{\mathcal{M}}(\bar{x})$. ■

We already know how to check for minimality. We now show how to convert a given SS to an equivalent minimal SS using the introduced transformations on SSs.

Proposition 13. The following procedure converts a given SS $\mathcal{M}$ to an equivalent trimmed and minimal SS $\mathcal{M}''$:

Algorithm 2: Minimization of a SS $\mathcal{M}$

- 1 Construct a basis $\{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in X\}$ for the state space W of $\mathcal{M}$
 $\Phi \leftarrow [\tau_{\bar{x}}\omega_\varepsilon]_{\bar{x} \in X}$
 $\mathcal{M}' \leftarrow \Phi^\dagger \mathcal{M} \Phi$
 - 2 Construct a basis $\{(\sigma'\tau_{\bar{y}}')^\top \mid \bar{y} \in Y'\}$ for the co-state space $\tilde{W}'$ of $\mathcal{M}'$
 $\Pi' \leftarrow [(\sigma'\tau_{\bar{y}}')^\top]_{\bar{y} \in Y'}^\top$
 $\mathcal{M}'' \leftarrow \Pi' \mathcal{M}' \Pi'^\dagger$
-

Proof. Note that by construction the columns of Φ and $\Pi'^\top$ form bases for the spaces W and $\tilde{W}'$, respectively. Therefore, $\Phi^\dagger \Phi = \mathbf{I}$ and $\Phi\Phi^\dagger|_W$ is the identity mapping, as

well as $(\Pi'^\top)^\dagger \Pi'^\top = \mathbf{I}$ and $\Pi'^\top (\Pi'^\top)^\dagger|_{\tilde{W}'}$ is the identity mapping. We can simply check equivalence, i.e., that for any $\bar{x} \in \Sigma^*$,

$$\begin{aligned}
f_{\mathcal{M}''}(\bar{x}) &= \sigma'' \tau''_{x_n} \cdots \tau''_{x_1} \omega''_\varepsilon \\
&= \sigma' \Pi'^\dagger \Pi' \tau'_{x_n} \Pi'^\dagger \cdots \Pi' \tau'_{x_1} \Pi'^\dagger \Pi' \omega'_\varepsilon \\
&= \omega'_\varepsilon{}^\top \Pi'^\top (\Pi'^\top)^\dagger \tau'_{x_1}{}^\top \Pi'^\top \cdots (\Pi'^\top)^\dagger \tau'_{x_n}{}^\top \Pi'^\top (\Pi'^\top)^\dagger \sigma'^\top \\
&= \sigma' \tau'_{x_n} \cdots \tau'_{x_1} \omega'_\varepsilon \\
&= \sigma \Phi \Phi^\dagger \tau_{x_n} \Phi \cdots \Phi^\dagger \tau_{x_1} \Phi \Phi^\dagger \omega_\varepsilon \\
&= \sigma \tau_{\bar{x}} \omega_\varepsilon = f_{\mathcal{M}}(\bar{x}).
\end{aligned}$$

Next, we observe $[\tau'_\bar{x} \omega'_\varepsilon]_{\bar{x} \in X} = [\Phi^\dagger \tau_{\bar{x}} \omega_\varepsilon]_{\bar{x} \in X} = \Phi^\dagger \Phi = \mathbf{I}$. This implies that $\mathcal{M}'$ has full state space W' and that $\{\tau'_\bar{x} \omega'_\varepsilon \mid \bar{x} \in X\}$ is a basis for W' , since the dimension d' of $\mathcal{M}'$ is $|X|$ by construction. By Proposition 9, $|X| = d'$ implies $\text{rank}(\mathcal{M}') = \min(|Y'|, |X|) = |Y'|$. By construction $|Y'| = d''$ where d'' is the dimension of $\mathcal{M}''$. Furthermore, $\text{rank}(\mathcal{M}') = \text{rank}(\mathcal{M}'')$ since $\mathcal{M}' \cong \mathcal{M}''$ so by Corollary 10 $\mathcal{M}''$ is trimmed and minimal. $\blacksquare$

As we can convert any SS to an equivalent minimal SS using the above Algorithm 2, it will be sufficient to characterize equivalence only for minimal SSs. This is done by the following result.

Proposition 14. *Let $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ and $\mathcal{M}' = (\sigma', \{\tau'_z\}, \omega'_\varepsilon)$ be minimal d -dimensional SS. The following are equivalent:*

- (i) $\mathcal{M} \cong \mathcal{M}'$
- (ii) $\mathcal{M}' = \rho \mathcal{M} \rho^{-1}$ for some non-singular $\rho \in K^{d \times d}$
- (iii) $\Pi \Phi = \Pi' \Phi'$, $\Pi \omega_\varepsilon = \Pi' \omega'_\varepsilon$, $\sigma \Phi = \sigma' \Phi'$ and $\forall z \in \Sigma : \Pi \tau_z \Phi = \Pi' \tau'_z \Phi'$, where $\{\tau_{\bar{x}} \omega_\varepsilon \mid \bar{x} \in X\}$ and $\{(\sigma \tau_{\bar{y}})^\top \mid \bar{y} \in Y\}$ are bases for the state and co-state spaces W and $\tilde{W}$ of $\mathcal{M}$, respectively, and $\Pi = [(\sigma \tau_{\bar{y}})^\top]_{\bar{y} \in Y}^\top$, $\Phi = [\tau_{\bar{x}} \omega_\varepsilon]_{\bar{x} \in X}$, $\Pi' = [(\sigma' \tau'_{\bar{y}})^\top]_{\bar{y} \in Y}^\top$, and $\Phi' = [\tau'_{\bar{x}} \omega'_\varepsilon]_{\bar{x} \in X}$.

Proof. Lemma 12 establishes (ii) $\Rightarrow$ (i). For (i) $\Rightarrow$ (iii) note that $f_{\mathcal{M}} = f_{\mathcal{M}'}$ implies that $\Pi \tau_{\bar{z}} \Phi = [f(\bar{x} \bar{z} \bar{y})]_{\bar{y} \in Y, \bar{x} \in X} = \Pi' \tau'_{\bar{z}} \Phi'$ for all $\bar{z} \in \Sigma^*$, as well as $\Pi \omega_\varepsilon = [f(\bar{y})]_{\bar{y} \in Y}^\top = \Pi' \omega'_\varepsilon$ and $\sigma \Phi = [f(\bar{x})]_{\bar{x} \in X} = \sigma' \Phi'$. Finally, to see (iii) $\Rightarrow$ (ii), note that Π and Φ have full rank, since $\mathcal{M}$ is minimal, so Π' and Φ' must also have full rank. Let $\rho = \Pi'^{-1} \Pi = \Phi' \Phi^{-1}$, since $\Pi \Phi = \Pi' \Phi'$, then $\rho^{-1} = \Phi \Phi'^{-1}$, and we can easily see that the conditions in (iii) imply $\mathcal{M}' = \rho \mathcal{M} \rho^{-1}$. $\blacksquare$

3. SEQUENTIAL SYSTEMS

Note that this allows us to decide equivalence for any given SSs $\mathcal{M}$ and $\mathcal{M}'$ by first converting them to equivalent minimal SSs $\tilde{\mathcal{M}}$ and $\tilde{\mathcal{M}}'$, respectively, using Algorithm 2 and then checking for equivalence by criteria (iii) from the above Proposition 14. The required bases for the state and co-state space of $\tilde{\mathcal{M}}$ or $\tilde{\mathcal{M}}'$ can be computed by Algorithm 1.

The following proposition shows that any SS – except for degenerate cases – can be transformed into an equivalent SS where σ and ω_ε can be essentially any desired vectors. This implies that it is no restriction to assume some fixed form for σ , as is sometimes done. For instance, in the case of OOMs often $\sigma = \mathbb{1}^\top$ is used, while for MA often $\sigma = [1, 0, \dots, 0]$ is assumed.

Proposition 15. *Let $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be a non-degenerate d -dimensional SS, by which we mean that $\sigma \neq 0$ and $\omega_\varepsilon \neq 0$, and select any $\sigma'^\top, \omega'_\varepsilon \in K^d \setminus \{0\}$ such that $\sigma' \omega'_\varepsilon = \sigma \omega_\varepsilon$. Then there exists a non-singular linear map ρ such that $\rho \mathcal{M} \rho^{-1} = (\sigma', \{\tau'_z\}, \omega'_\varepsilon)$.*

Proof. Extend $\{\sigma^\top\}$ to an orthogonal basis $\{\sigma^\top, v_2, \dots, v_d\}$ of K^d , and $\{\sigma'^\top\}$ to an orthogonal basis $\{\sigma'^\top, v'_2, \dots, v'_d\}$ of K^d , and define $e_1 = [1, 0, \dots, 0]^\top$ and $e_2 = [0, 1, 0, \dots, 0]^\top$. We distinguish two cases:

If $c := \sigma \omega_\varepsilon = \sigma' \omega'_\varepsilon \neq 0$, then $\rho_1 = [\omega_\varepsilon, v_2, \dots, v_d]$ and $\rho_2 = [\omega'_\varepsilon, v'_2, \dots, v'_d]$ are non-singular. Let $\rho = \rho_2 \rho_1^{-1}$. We can easily see that $\rho_2 \rho_1^{-1} \omega_\varepsilon = \rho_2 e_1 = \omega'_\varepsilon$ and $\sigma \rho^{-1} = \sigma \rho_1 \rho_2^{-1} = c \cdot e_1^\top \rho_2^{-1} = \sigma'$, since $\sigma' \rho_2 = c \cdot e_1^\top$.

If $\sigma \omega_\varepsilon = \sigma' \omega'_\varepsilon = 0$, then $\rho_1 = [\frac{\sigma^\top}{\sigma \sigma^\top}, \omega_\varepsilon, v_3, \dots, v_d]$ and $\rho_2 = [\frac{\sigma'^\top}{\sigma' \sigma'^\top}, \omega'_\varepsilon, v'_3, \dots, v'_d]$ (perhaps after first reordering the v_i and v'_i) are non-singular. Let $\rho = \rho_2 \rho_1^{-1}$. We can again check that $\rho_2 \rho_1^{-1} \omega_\varepsilon = \rho_2 e_2 = \omega'_\varepsilon$ and $\sigma \rho^{-1} = \sigma \rho_1 \rho_2^{-1} = e_1^\top \rho_2^{-1} = \sigma'$, since $\sigma' \rho_2 = e_1^\top$. ■

Finally, we introduce a special property called *interpretability* that a SS can have. This concept has led to some confusion in the past – especially regarding the relationship between OOMs and PSRs. This is due to the fact that it has been defined differently for OOMs, IO-OOMs, and PSRs, as will be discussed later. Another source of confusion is that interpretability has been regarded as a crucial property for learning, which is, however, only true for the very early learning algorithms. Here we give a definition of interpretability that works for all models, and we will defer the discussion of the different uses to the later sections.

Definition 16. *A d -dimensional SS $\mathcal{M}$ is interpretable w.r.t. the sets $Y_1, \dots, Y_d \subset \Sigma^*$ if the states $\omega_{\bar{x}}$ take the form $\omega_{\bar{x}} = [f_{\mathcal{M}}(\bar{x}Y_1), \dots, f_{\mathcal{M}}(\bar{x}Y_d)]^\top$ for all $\bar{x} \in \Sigma^*$, where $f_{\mathcal{M}}(\bar{x}Y) = \sum_{\bar{y} \in Y} f_{\mathcal{M}}(\bar{x}\bar{y})$.*

The following proposition and algorithm show how to *make a SS interpretable*, i.e., how to convert any given SS into an equivalent interpretable form.

Proposition 17. *Let $Y_1, \dots, Y_d \subset \Sigma^*$ and $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be a d -dimensional minimal SS. If $\rho = [(\sigma\tau_{Y_1})^\top, \dots, (\sigma\tau_{Y_d})^\top]^\top$ is non-singular, where $\tau_Y = \sum_{\bar{y} \in Y} \tau_{\bar{y}}$, then $\mathcal{M}'$ defined as $\rho\mathcal{M}\rho^{-1}$ is interpretable w.r.t. $Y_1, \dots, Y_d$ and equivalent to $\mathcal{M}$.*

Proof. $\forall \bar{x} \in \Sigma^* : \omega'_{\bar{x}} = \rho\omega_{\bar{x}} = [\sigma\tau_{Y_1}\tau_{\bar{x}}\omega_\varepsilon, \dots, \sigma\tau_{Y_d}\tau_{\bar{x}}\omega_\varepsilon]^\top = [f_{\mathcal{M}}(\bar{x}Y_1), \dots, f_{\mathcal{M}}(\bar{x}Y_d)]^\top$. ■

Corollary 18. *For a SS $\mathcal{M}$, Algorithm 3 returns an equivalent interpretable SS.*

Algorithm 3: Make a SS $\mathcal{M}$ of rank d interpretable

- 1 Minimize $\mathcal{M}$, i.e., find an equivalent minimal SS $\mathcal{M}'$ using Algorithm 2
 - 2 Find a basis $\{(\sigma'\tau'_{\bar{y}})^\top \mid \bar{y} \in Y\}$ of the co-state space $\tilde{W}'$ of $\mathcal{M}'$ using Algorithm 1
 - 3 $\rho \leftarrow [(\sigma'\tau'_{\bar{y}})^\top]_{\bar{y} \in Y}^\top$
 - 4 **return** $\rho\mathcal{M}'\rho^{-1}$
-

Proof. By Proposition 17 the above algorithm returns an equivalent SS that is interpretable w.r.t. the singleton sets $Y_k = \{\bar{y}_k\}$, where $\{\bar{y}_1, \dots, \bar{y}_d\} = Y$ is an enumeration of Y , since $\mathcal{M}'$ is minimal and therefore ρ is non-singular by construction. ■

3. SEQUENTIAL SYSTEMS

4

Versions of Sequential Systems

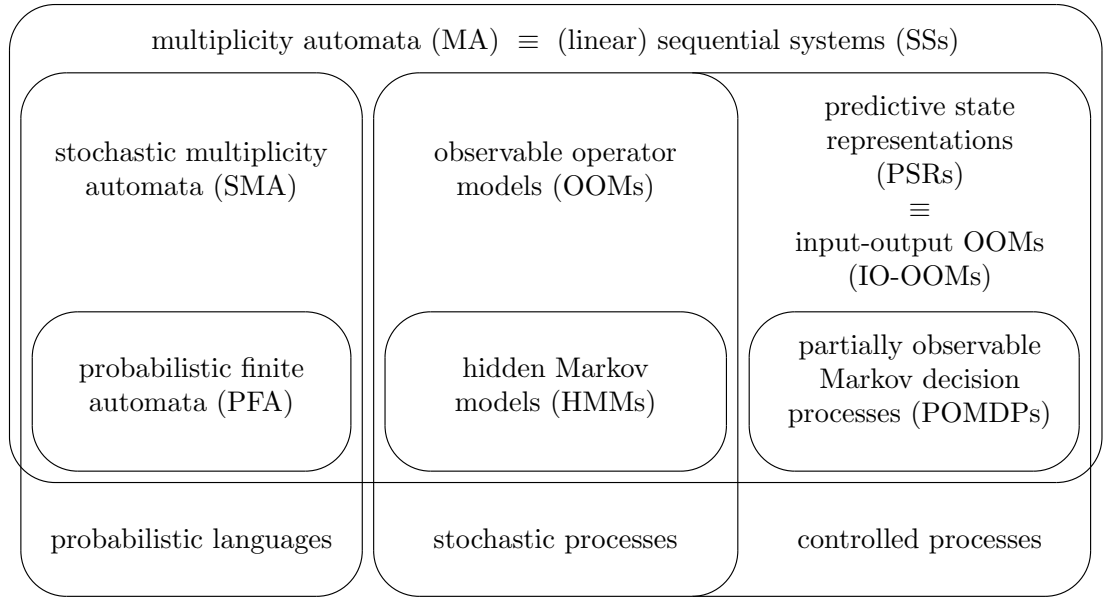


Figure 1: SMA, OOMs, and PSRs are versions of SSs that model probabilistic languages, stochastic processes, and controlled processes, respectively, and strictly generalize PFA, HMMs, and POMDPs, respectively.

In this chapter we first show that sequential systems (SSs) are an algebraic characterization of multiplicity automata (MA), and we mention the relationship to the more general class of weighted finite automata (WFA) and its extension to weighted finite-state transducers (WFST). We then define stochastic multiplicity automata (SMA), observable operator models (OOMs), and predictive state representations (PSRs), which are known to generalize probabilistic finite automata (PFA), hidden Markov models

4. VERSIONS OF SEQUENTIAL SYSTEMS

(HMMs), and partially observable Markov decision processes (POMDPs), respectively. We show that these are all instances of SSs that are used to model different kinds of objects. Furthermore, we examine the relationships between these models. An overview is given in Figure 1.

4.1 Multiplicity Automata and Weighted Automata

The above Definition 1 of linear finite dimensional SSs is an equivalent algebraic way of looking at a type of automata that were introduced by Schützenberger [1961] and are most commonly known as *multiplicity automata* [Salomaa and Soittola, 1978; Berstel and Reutenauer, 1988]. We will give a very brief introduction.

Definition 19. A K -multiplicity automaton (MA) is a structure $\langle \Sigma, Q, \varphi, \iota, \tau \rangle$, where Σ is an alphabet, Q is a finite set of states, $\varphi : Q \times \Sigma \times Q \rightarrow K$ is the state transition function, $\iota : Q \rightarrow K$ is the initialization function, and $\tau : Q \rightarrow K$ is the termination function. The state transition function is extended to words by setting $\forall \bar{x} \in \Sigma^*, z \in \Sigma : \varphi(q, \bar{x}z, q') = \sum_{s \in Q} \varphi(q, \bar{x}, s) \varphi(s, z, q')$, as well as $\varphi(q, \varepsilon, q') = 1$ if $q = q'$ and 0 otherwise. A multiplicity automaton $\mathcal{M}$ then defines a function

$$f_{\mathcal{M}} : \Sigma^* \rightarrow K, \quad f_{\mathcal{M}}(\bar{x}) = \sum_{q, q' \in Q} \iota(q) \varphi(q, \bar{x}, q') \tau(q').$$

The formal equivalence of MA to linear finite-dimensional SSs is easily seen by rewriting the definition of MA in terms of matrix multiplication:

Set $\omega_{\varepsilon} = [\iota(q)]_{q \in Q}^{\top}$, $\tau_z = [\varphi(q, z, q')]_{q' \in Q, q \in Q}$, and $\sigma = [\tau(q')]_{q' \in Q}$. We can show by induction on the length of $\bar{x}$ that this implies $\tau_{\bar{x}} = [\varphi(q, \bar{x}, q')]_{q' \in Q, q \in Q}$ for all $\bar{x} \in \Sigma^*$:

$$\begin{aligned} \tau_{\bar{x}z} &= \tau_z \tau_{\bar{x}} \\ &= [\varphi(q'', z, q')]_{q' \in Q, q'' \in Q} [\varphi(q, \bar{x}, q'')]_{q'' \in Q, q \in Q} \\ &= \left[\sum_{q'' \in Q} \varphi(q, \bar{x}, q'') \varphi(q'', z, q') \right]_{q' \in Q, q \in Q} \\ &= [\varphi(q, \bar{x}z, q')]_{q' \in Q, q \in Q} \end{aligned}$$

Therefore, for all $\bar{x} \in \Sigma^*$, we get $\sigma \tau_{\bar{x}} \omega_{\varepsilon} = [\tau(q')]_{q' \in Q} [\varphi(q, \bar{x}z, q')]_{q' \in Q, q \in Q} [\iota(q)]_{q \in Q}^{\top} = \sum_{q, q' \in Q} \tau(q') \varphi(q, \bar{x}, q') \iota(q) = f_{\mathcal{M}}(\bar{x})$.

However, the above definition of MA makes it apparent how MA are an extension of non-deterministic finite automata (NFA) to WFA that add weights to the initial

4.1 Multiplicity Automata and Weighted Automata

and terminal states as well as the state transitions. The weight of a path from an initial state to a termination state is then given by the product of the corresponding weights (hence the name *multiplicity* automata), while the value $f_{\mathcal{M}}(\bar{x})$ is computed by summing the weights of all paths compatible with $\bar{x}$.

At this point we should mention that MA as defined here are merely a special case of WFA. The difference is that for MA we consider weights from a field K (here $K = \mathbb{R}$ or $K = \mathbb{C}$), while for WFA the weights are only required to come from an algebraic structure K called a semiring. A large body of theory for WFA that generalizes the theory of SSs that we have presented in Chapter 3 can be found in the recent textbook by Droste et al. [2009]. Note that while MA and WFA are formally closely related, there is a difference in the way they are viewed and used. For instance, WFA are often considered over the semiring $\mathbb{R}^+$ with weights given the interpretation of transition probabilities, which are then called probabilistic finite automata (PFA). Such PFA are graphical models, and the states Q are latent states. For $\mathbb{R}$ -MA, however, the weights are allowed to be negative, and the weights as well as the states Q become abstract notions. In other words, PFA (and WFA in general) are typically used when the states and transition structure carry some meaning, while MA are typically used as an abstract tool to characterize functions $f : \Sigma^* \rightarrow K$. This difference in perspective is reflected in the relationship of PFA to SMA, HMM to OOM, and POMDP to PSR described in the remainder of this Chapter 4. Note that PFA are a special case of MA, as $\mathbb{R}^+ \subset \mathbb{R}$. In fact, there exist functions $f : \Sigma^* \rightarrow \mathbb{R}^+$ that can be described by a MA but not by a PFA, i.e., MA are strictly more general than PFA. This sequence of increasing generalization starting with finite automata (FA) can be summarized as follows:

$$\text{FA} \subset \text{NFA} \subset \text{PFA} \subset \text{MA} \equiv \text{SS} \subset \text{WFA}.$$

Furthermore, there exists a natural extension of WFA to input-output systems that are called weighted finite-state transducers (WFST). Here, the alphabet Σ is split as $\Sigma = \Sigma_I \times \Sigma_O$, where Σ_I is regarded as input alphabet and Σ_O as output alphabet. The function $f_{\mathcal{M}} : \Sigma_I^* \times \Sigma_O^* \rightarrow K$ is then viewed as describing a relation between Σ_I and Σ_O . Again, K is in general only required to be a semiring, but a typical choice is $K = \mathbb{R}^+$ with the interpretation of state transition probabilities, yielding a latent variable model called probabilistic finite-state transducers (PFST). WFST are a flexible class of models that have been shown to unify several common approaches used in the context of language and speech processing; a survey is given by Mohri et al. [2002]. Furthermore, IO-OOMs and thereby PSRs (see Section 4.2 and Section 4.3) are in

4. VERSIONS OF SEQUENTIAL SYSTEMS

fact WFST with weights in $K = \mathbb{R}$, although they are not usually viewed this way, as WFST are typically seen as latent variable models, while IO-OOMs and PSRs are not. However, since PFST are MA, as long as the desired application merely requires the characterization of the function $f_{\mathcal{M}} : \Sigma_I^* \times \Sigma_O^* \rightarrow \mathbb{R}^+$, the SS learning algorithms described in Chapter 5 can be applied to the case of WFST as well, as has been done recently by Balle et al. [2011].

Note that in the context of MA one is usually interested in characterizing functions $f : \Sigma^* \rightarrow K$, which are also called *formal series* in general and *recognizable series* if they are computed by a MA. However, a MA $\mathcal{M}$ can also be used to recognize a language $L \subseteq \Sigma^*$ by setting $L_{\mathcal{M}} = \{\bar{x} \in \Sigma^* \mid f_{\mathcal{M}}(\bar{x}) \subseteq J\}$ for some subset $J \subseteq K$, e.g. $J = \{k \in K : k > \kappa\}$ for some threshold parameter $\kappa \geq 0$. The class of languages recognizable by MA is known to be strictly more general than the class of regular languages [Cortes and Mohri, 2000].

MA have received a lot of attention in the context of learning theory following the discovery of efficient learning algorithms [Bergadano and Varricchio, 1994; Ohnishi et al., 1994] in an extended version of the exact learning model of Angluin [1987]. This led to further results on the learnability of several classes of DNF formulas [Bergadano et al., 1996], the class of polynomials over finite fields, decision trees, and others [Beimel et al., 1996, 2000].

4.1.1 Stochastic Multiplicity Automata and Stochastic Languages

Additionally, MA have been applied in the context of probabilistic grammatical inference [Denis et al., 2006; Bailly et al., 2009], which is of particular interest to us because of the close relationship of these approaches to OOMs and PSRs, as we shall see.

Definition 20. A function $f : \Sigma^* \rightarrow \mathbb{R}$ that satisfies $0 \leq f \leq 1$ and $f(\Sigma^*) = \sum_{\bar{x} \in \Sigma^*} f(\bar{x}) = 1$ is called a *stochastic language*, *probabilistic language* or *just distribution over Σ^** . A distribution $f_{\mathcal{M}}$ over Σ^* that is defined by some MA $\mathcal{M}$ is called a *rational stochastic language*, and a MA that defines such a distribution is called a *stochastic MA (SMA)*.

Denis and Esposito [2008] give a comprehensive overview of rational stochastic languages over various fields K as well as their relationships to each other and to subclasses such as the important class of probabilistic regular languages.

Definition 21. A probabilistic (finite) automaton (PFA) is a SMA with the following restrictions: (i) ι, τ, φ must have values in $[0, 1]$, and (ii) $\iota(Q) = 1$ and $\forall q \in Q :$

4.2 Observable Operator Models and Stochastic Processes

$\tau(q) + \varphi(q, \Sigma, Q) = 1$, where $\iota(Q) = \sum_{q \in Q} \iota(q)$ and $\varphi(q, \Sigma, Q) = \sum_{x \in \Sigma} \sum_{q' \in Q} \varphi(q, x, q')$. The stochastic languages that can be represented by PFA are called probabilistic regular languages.

PFA are closely related to HMMs, and the relationship has been described in detail by Dupont et al. [2005]. It is, however, less well known that SMA are closely related to OOMs, which generalize HMMs in a similar way that SMA generalize PFA.

We point out two results that are relevant in the context of modeling probabilistic languages by MA. First of all, it is known that it is an NP-hard problem to compute the maximum likelihood estimate of parameters of a PFA with known structure from a given training set of words [Abe and Warmuth, 1992]. In practice, algorithms based on expectation maximization (EM) [Dempster et al., 1977] are used which compute locally optimal models instead. In contrast to this, the algebraic theory for SSs allows for powerful learning algorithms (see Chapter 5) that often outperform EM-trained PFA or HMMs [Rosencrantz et al., 2004; Jaeger et al., 2006a]. However, these learning algorithms may return MA that are arbitrarily close to SMA but fail to represent stochastic languages. It is, in fact, undecidable whether a MA represents a stochastic language [Denis and Esposito, 2004].

4.2 Observable Operator Models and Stochastic Processes

OOMs were introduced by Jaeger [1997] as a concise algebraic characterization of stochastic processes [see also Jaeger, 1998, 2000b; Jaeger et al., 2006b]. These models are closely related to other algebraic characterizations of stochastic processes [Heller, 1965; Ito, 1992; Upper, 1997] that were studied in the context of deciding the equivalence for HMMs [Gilbert, 1959], which came to a successful conclusion by framing HMMs in the more general class of *linearly dependent processes* by Ito et al. [1992].

Definition 22. A (Σ -valued) stochastic process is a function $f : \Sigma^* \rightarrow [0, 1]$ that satisfies (i) $f(\varepsilon) = 1$ and (ii) $\forall \bar{x} \in \Sigma^* : f(\bar{x}) = \sum_{z \in \Sigma} f(\bar{x}z)$. An observable operator model (OOM) is a linear SS $\mathcal{M}$ over the field $K = \mathbb{R}$ such that $f_{\mathcal{M}}$ is a stochastic process. A stochastic process that can be modeled by a finite dimensional OOM is called a linearly dependent process.

Note that a stochastic process function f as defined above defines a Σ -valued stochastic process $\{X_t\}_{t \in \mathbb{N}}$ in the usual mathematical sense by specifying its *finite-dimensional distributions* via $\mathbb{P}(X_1 = x_1, \dots, X_l = x_l) = \mathbb{P}(\bar{x}) = f(\bar{x})$ for all $\bar{x} \in \Sigma^l$

4. VERSIONS OF SEQUENTIAL SYSTEMS

and $l \in \mathbb{N}$. Existence and uniqueness of the corresponding stochastic process $\{X_t\}_{t \in \mathbb{N}}$ are guaranteed by the Kolmogorov extension theorem.

One of the interesting features of OOMs is their notion of “state” of a (stochastic) process. The idea that goes back to Zadeh [1969] is that a system state is really nothing more than the information that is required to predict the future. In the case of OOMs, the states $\omega_{\bar{x}}$ not only carry enough information to predict the future, they *are* (in a certain sense) just future predictions.

To see this, recall that the states $\omega_{\bar{x}}$ of a SS are coordinate representations of the functions $f_{\bar{x}}$ w.r.t. some unknown basis $\mathcal{B}$ of the function space $\mathcal{F}$. In the case of OOMs, these functions take on the meaning that $f_{\bar{x}}(\bar{y}) = \mathbb{P}(\bar{x}\bar{y})$, i.e., they give the probability of observing the sequence $\bar{x}$ followed by $\bar{y}$. These functions are therefore called *future prediction functions* in the context of OOMs. The operators $\{\tau_z\}$ are then state update operators that update a state $\omega_{\bar{x}}$ (corresponding to the future prediction function $f_{\bar{x}}$ after an initial observation of $\bar{x}$) according to the new observation z to the new state $\omega_{\bar{x}z}$ (corresponding to the future prediction function $f_{\bar{x}z}$ after an initial observation of $\bar{x}z$) – hence the name “observable operators” [Jaeger, 1998].

For convenience, these functions $f_{\bar{x}}$, as well as the corresponding states $\omega_{\bar{x}}$, are often normalized to $f_{\bar{x}}/f(\bar{x})$ and $\omega_{\bar{x}}/\sigma\omega_{\bar{x}}$, respectively, since $f_{\bar{x}}(\bar{y})/f(\bar{x}) = \sigma\tau_{\bar{y}}\omega_{\bar{x}}/\sigma\omega_{\bar{x}} = \mathbb{P}(\bar{y}|\bar{x})$, the probability of observing $\bar{y}$ given that $\bar{x}$ has been observed. Therefore, an OOM started in the normalized state $\omega_{\bar{x}}/\sigma\omega_{\bar{x}}$ represents a stochastic process started after an initial observation of $\bar{x}$. This corresponds to the notion of a *residual automaton* in the context of SMA, which is obtained by starting a SMA in the (normalized) state $\omega_{\bar{x}}/\sum_{\bar{z} \in \Sigma^*} \sigma\tau_{\bar{z}}\omega_{\bar{x}}$ and then represents a *residual language* [Denis and Esposito, 2004].

4.2.1 Relation to Hidden Markov Models

Any HMM can be trivially converted into an OOM. A HMM consists of an unobserved Markov process Z_t that takes values in a finite set Q of states and that is governed by a state transition matrix $T = [\mathbb{P}(Z_{t+1} = q' | Z_t = q)]_{q \in Q, q' \in Q}$. At each time step an observation X_t from Σ is made according to the probabilities given in the emission vectors $E_x = [\mathbb{P}(X_t = x | Z_t = q)]_{q \in Q}^\top$ and the state q at time t . Finally, an initial state vector $\pi = [\mathbb{P}(Z_1 = q)]_{q \in Q}^\top$ is needed to fully specify the distribution of the stochastic process Y_t [Rabiner, 1989].

Proposition 23. [Jaeger, 2000b] *A given HMM $(T, \{E_x\}_{x \in \Sigma}, \pi)$ with d states is equivalent to the d -dimensional OOM $(\sigma, \{\tau_x\}, \omega_\varepsilon)$ defined by $\sigma = \mathbb{1}_d^\top$, $\tau_x = T^\top \mathbf{D}(E_x)$ and $\omega_\varepsilon = \pi$. The rank of the OOM is less than or equal to d .*

Moreover, there are examples of OOMs of finite rank that cannot be modeled by any HMM with a finite number of states. A prototypical example is the so-called “probability clock” [Jaeger, 1998]. It is still an open question how to find a “close” HMM for a given OOM in general, but under certain conditions this is indeed possible [Hsu et al., 2009; Anandkumar et al., 2012]. While OOMs can be seen as a generalization of HMMs, one should keep in mind that there is a fundamental difference in the notion of the state of the process. The state vector in the case of a HMM is a stochastic vector that expresses the belief about the underlying hidden state, while for an OOM it is a coordinate representation of the corresponding future prediction function.

4.2.2 Relation to Stochastic Multiplicity Automata

The main difference between OOMs and SMA is that OOMs model stochastic processes, while SMA model distributions on words. However, we can use a stochastic process to model a distribution on words if we introduce a termination symbol \$.

Definition 24. An OOM $\mathcal{M}$ over the alphabet $\Sigma_\$ = \Sigma \cup \{\$\}$ is said to be terminating if $f_{\mathcal{M}}(\Sigma^*\$) := \sum_{\bar{x} \in \Sigma^*} f_{\mathcal{M}}(\bar{x}\$) = 1$.

Definition 25. A terminating OOM $\mathcal{M}$ over the alphabet $\Sigma \cup \{\$\}$ and a SMA $\mathcal{A}$ over the alphabet Σ are related, if $f_{\mathcal{M}}(\bar{x}\$) = f_{\mathcal{A}}(\bar{x})$ for all $\bar{x} \in \Sigma^*$.

Note that a SMA $\mathcal{A}$ and a related terminating OOM $\mathcal{M}$ contain the same information about the function $f_{\mathcal{A}}(\bar{x})$. Next, we show how to construct a related terminating OOM $\mathcal{M}$ for any given SMA $\mathcal{A}$.

Lemma 26. If $\mathcal{A} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ is a minimal d -dimensional SMA, then $\tau_{\Sigma^*} = \sum_{k=0}^{\infty} \tau_{\Sigma}^k$ exists and is equal to $(\mathbf{I}_d - \tau_{\Sigma})^{-1}$, where $\tau_{\Sigma} = \sum_{z \in \Sigma} \tau_z$.

Proof. We will show that the spectral radius¹ $\rho(\tau_{\Sigma})$ satisfies $\rho(\tau_{\Sigma}) < 1$, which implies the lemma. Assume $\rho(\tau_{\Sigma}) \geq 1$, i.e., there exists some $\lambda \in \mathbb{C}, |\lambda| \geq 1$ and $v \in \mathbb{C}^d$ such that $\tau_{\Sigma}v = \lambda v$. As $\mathcal{A}$ is minimal, we may find sets of words $X, Y \subset \Sigma^*$ such that $\Pi = [(\sigma\tau_{\bar{y}})^{\top}]_{\bar{y} \in Y}^{\top}$ and $\Phi = [\tau_{\bar{x}}\omega_\varepsilon]_{\bar{x} \in X}$ with $|X| = |Y| = d$ are non-singular using Algorithm 1. Then $v = \Phi a$ for some $a \in \mathbb{C}^d$, and $\Pi\tau_{\Sigma}^k\Phi a = \lambda^k\Pi\Phi a$ for any $k \in \mathbb{N}$. Now the SMA property $f_{\mathcal{A}}(\Sigma^*) = \sum_{k=0}^{\infty} \sigma\tau_{\Sigma}^k\omega_\varepsilon = 1$ implies that $\Pi\tau_{\Sigma}^k\Phi \rightarrow 0$ as $k \rightarrow \infty$, while the right hand side $\lambda^k\Pi\Phi a$ does not (note $\Pi\Phi a \neq 0$), which is a contradiction. ■

¹For $A \in \mathbb{C}^{n \times n}$ with eigenvalues $\lambda_1, \dots, \lambda_k$, the spectral radius is defined as $\rho(A) = \max_i |\lambda_i|$.

4. VERSIONS OF SEQUENTIAL SYSTEMS

Proposition 27. *Let $\mathcal{A} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be a minimal d -dimensional SMA. Then $\mathcal{M} = (\sigma', \{\tau'_z\}, \omega'_\varepsilon)$ is a related $(d+1)$ -dimensional terminating OOM over the alphabet $\Sigma_\$ = \Sigma \cup \{\$\}$, if*

- $\sigma' = \left[\sigma \sum_{k=0}^{\infty} \tau_\Sigma^k, \ 1 \right] = \left[\sigma(\mathbf{I}_d - \tau_\Sigma)^{-1}, \ 1 \right],$
- $\tau'_z = \begin{bmatrix} \tau_z & 0 \\ 0 & 0 \end{bmatrix}, \quad \tau'_\$ = \begin{bmatrix} 0 & 0 \\ \sigma & 1 \end{bmatrix}, \text{ and}$
- $\omega'_\varepsilon = \begin{bmatrix} \omega_\varepsilon \\ 0 \end{bmatrix}.$

Proof. We can simply check that for all $\bar{z} \in \Sigma_\$^*$

$$f_{\mathcal{M}}(\bar{z}) = \sigma' \tau'_{\bar{z}} \omega'_\varepsilon = \begin{cases} \sigma(\sum_{k=0}^{\infty} \tau_\Sigma^k) \tau_{\bar{z}} \omega_\varepsilon & \text{if } \bar{z} \in \Sigma^*, \\ \sigma \tau_{\bar{x}} \omega_\varepsilon & \text{if } \bar{z} = \bar{x}\$ \dots \$ \text{ for some } \bar{x} \in \Sigma^*, \\ 0 & \text{otherwise.} \end{cases}$$

This implies $f_{\mathcal{M}} \geq 0$, $f_{\mathcal{M}}(\bar{x}\$) = f_{\mathcal{A}}(\bar{x})$ for all $\bar{x} \in \Sigma^*$ ($\mathcal{M}$ and $\mathcal{A}$ are related), as well as $f_{\mathcal{M}}(\Sigma^*\$) = f_{\mathcal{A}}(\Sigma^*) = 1$ ($\mathcal{M}$ is terminating if it is an OOM). Furthermore, $\sigma' \omega'_\varepsilon = f_{\mathcal{A}}(\Sigma^*) = 1$ and $\sigma' \tau'_{\Sigma\$} = [\sigma \sum_{k=0}^{\infty} \tau_\Sigma^k \tau_\Sigma + \sigma, 1] = \sigma'$, which imply property (i) and (ii) for a stochastic process, respectively. $\blacksquare$

The conversion given in Proposition 27 of a SMA to a related terminating OOM is important for two reasons. First of all, as noted by Wiewiora [2008] using a slightly different conversion, this can be used to show that determining if a SS is an OOM is an undecidable problem, as any such decision procedure could be used to construct a procedure that determines whether a SS is a SMA essentially via this conversion, which is known to be an undecidable problem. Secondly, it allows us to use the normalized sampling procedure that we present in Section 5.6 to deal with approximate OOMs also for the case of approximate SMA.

Conversely, a stochastic process $f_{\mathcal{M}}$ modeled by an OOM $\mathcal{M}$ can be described by a SMA that is related to a terminating version of $\mathcal{M}$ as follows.

Proposition 28. *An OOM $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ over the alphabet Σ can be extended to a terminating OOM $\mathcal{M}' = (\sigma, \{\tau'_z\}, \omega_\varepsilon)$ over the alphabet $\Sigma_\$ = \Sigma \cup \{\$\}$, such that $f_{\mathcal{M}}(\bar{x}) = p^{-1}(1-p)^{-|\bar{x}|} f_{\mathcal{M}'}(\bar{x}\$)$ for all $\bar{x} \in \Sigma^*$, by setting $\tau'_z = (1-p)\tau_z$ and $\tau'_\$ = p\tau_\Sigma$ for some fixed termination probability $p \in (0, 1)$, where $\tau_\Sigma = \sum_{z \in \Sigma} \tau_z$.*

Proof. We first show that $\mathcal{M}'$ describes a stochastic process. Clearly, $f_{\mathcal{M}'} \geq 0$ and $f_{\mathcal{M}'}(\varepsilon) = \sigma\omega_\varepsilon = 1$. To show property (ii), take any $\bar{x} \in \Sigma_\$^*$ and note that by linearity $\tau'_{\bar{x}}\omega_\varepsilon = \sum_k \lambda_k \tau_{\bar{x}_k}\omega_\varepsilon$ for suitable $\lambda_k \in \mathbb{R}$ and sequences $\bar{x}_k \in \Sigma^*$ (this is obtained by replacing all occurrences of $\tau'_\$$ by $p \sum_{z \in \Sigma} \tau_z$). Then $\sum_{z \in \Sigma_\$} f_{\mathcal{M}'}(\bar{x}z) = \sigma(\sum_{z \in \Sigma_\$} \tau'_z)\tau'_{\bar{x}}\omega_\varepsilon = \sigma\tau_\Sigma\tau'_{\bar{x}}\omega_\varepsilon = \sum_k \lambda_k \sigma\tau_\Sigma\tau_{\bar{x}_k}\omega_\varepsilon = \sum_k \lambda_k \sigma\tau_{\bar{x}_k}\omega_\varepsilon = \sigma\tau'_{\bar{x}}\omega_\varepsilon = f_{\mathcal{M}'}(\bar{x})$. Moreover, $f_{\mathcal{M}'}(\bar{x}\$) = \sigma\tau'_\$\tau'_{\bar{x}}\omega_\varepsilon = \sigma p\tau_\Sigma(1-p)^{|\bar{x}|}\tau_{\bar{x}}\omega_\varepsilon = p(1-p)^{|\bar{x}|} \sum_{z \in \Sigma} f_{\mathcal{M}}(\bar{x}z) = p(1-p)^{|\bar{x}|} f_{\mathcal{M}}(\bar{x})$ for all $\bar{x} \in \Sigma^*$, and therefore $f_{\mathcal{M}'}(\Sigma^*\$) = \sum_{\bar{x} \in \Sigma^*} f_{\mathcal{M}'}(\bar{x}\$) = \sum_{l=0}^{\infty} p(1-p)^l \sum_{\bar{x} \in \Sigma^l} f_{\mathcal{M}}(\bar{x}) = \sum_{l=0}^{\infty} p(1-p)^l = 1$. ■

Proposition 29. *Let $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be a d -dimensional terminating OOM over the alphabet $\Sigma \cup \{\$\}$. Then $\mathcal{A} = (\sigma\tau_\$, \{\tau_z\}, \omega_\varepsilon)$ is a related d -dimensional SMA over the alphabet Σ .*

Proof. Clearly, $f_{\mathcal{A}}(\bar{x}) = f_{\mathcal{M}}(\bar{x}\$) \geq 0$ for all $\bar{x} \in \Sigma^*$ and $f_{\mathcal{A}}(\Sigma^*) = f_{\mathcal{M}}(\Sigma^*\$) = 1$. ■

4.2.3 Historical Remarks

Note that our definition of OOMs given in Definition 22 differs slightly from the definition typically found in the literature.

First of all, the property (ii) for a stochastic process means that an OOM must satisfy $\sigma\tau_\Sigma\omega_{\bar{x}} = \sigma\omega_{\bar{x}}$ for all $\bar{x} \in \Sigma^*$, which implies (ii)' $\sigma\tau_\Sigma = \sigma$ if the OOM is minimal, but not in general. The property (ii)' is, however, often stated as part of the definition for OOMs. Our above Definition 22 is therefore slightly more relaxed than the standard definition in the case of non-minimal models, but this has no practical consequences.

Furthermore, for purely historical reasons, OOMs are sometimes required to satisfy $\sigma = \mathbb{1}_d^\top$. This is mainly an issue of normalization, and we have shown in Proposition 15 that any OOM can indeed be normalized such that $\sigma = \mathbb{1}_d^\top$. However, this in turn has led to a more restrictive definition of interpretability for OOMs. The point is that due to property (i) of stochastic processes, an OOM that satisfies $\sigma = \mathbb{1}_d^\top$ can only be interpretable with respect to sets $Y_1, \dots, Y_d$ if it also satisfies $1 = \sigma\omega_\varepsilon = \mathbb{1}_d^\top [f_{\mathcal{M}}(Y_k)]_{k \in \{1, \dots, d\}}^\top = \sum_{k=1}^d \sum_{\bar{y} \in Y_k} f_{\mathcal{M}}(\bar{y})$. This is typically assured by requiring the sets Y_k to partition Σ^l for some l . One can relax this restriction on the sets Y_k for the definition of interpretability – as we have done in Definition 16 – if one is willing to drop the normalization requirement $\sigma = \mathbb{1}_d^\top$ as well.

Nevertheless, even though the normalization requirement $\sigma = \mathbb{1}_d^\top$ is superfluous, several of the OOM learning algorithms have been designed to yield OOMs normalized such that $\sigma = \mathbb{1}_d^\top$ – oftentimes unnecessarily complicating the algorithms – and some proofs have made use of this normalization as well. Later in Sections 5.4.5 and 5.4.6

4. VERSIONS OF SEQUENTIAL SYSTEMS

we present simplified and generalized versions of the EC and ES learning algorithms by removing this normalization restriction from the algorithms and proofs.

4.3 Predictive State Representations and Controlled Processes

Following the development of OOMs for stochastic processes, extensions to the case of controlled processes – stochastic processes that depend on an external input at each time step – were proposed by Jaeger [1998] as *input-output* OOMs, by Littman et al. [2001] as *predictive state representations* (PSRs), and as a further variant as *transformed PSRs* by Rosencrantz et al. [2004]. All approaches are (in the linear case) equivalent and can be easily understood in the framework of linear SSs.

Definition 30. A (discrete-valued) controlled (stochastic) process *with input from* Σ_I *and output in* Σ_O *is a function* $f : \Sigma^* \rightarrow [0, 1]$ *that satisfies* (i) $f(\varepsilon) = 1$ *and* (ii) $\forall \bar{x} \in \Sigma^*, a \in \Sigma_I : f(\bar{x}) = \sum_{o \in \Sigma_O} f(\bar{x}ao)$, *where* $\Sigma = \Sigma_I \times \Sigma_O$. *We define* $f(\bar{y} | \bar{x}) = f(\bar{x}\bar{y})/f(\bar{x})$ *for* $f(\bar{x}) \neq 0$ *and zero otherwise. An input-output OOM (IO-OOM) is just a SS that models a controlled process.*

Note that the values of f are not probabilities. One may interpret $f(a_1 o_1 \dots a_n o_n)$ as “ $\mathbb{P}(o_1 \dots o_n | a_1 \dots a_n)$ ”, i.e., as the conditional probability of observing the outputs $o_1 \dots o_n$ given the inputs $a_1 \dots a_n$. However, one must take care, as the sequence of inputs may depend on the observed outputs as well. This is explained in more detail in Section 5.1.

Definition 31. Let f be a controlled process with predictive states $\dot{\omega}_{\bar{h}}$ defined as $\dot{\omega}_{\bar{h}} = [f(\bar{q}_1 | \bar{h}), \dots, f(\bar{q}_d | \bar{h})]^\top \in \mathbb{R}^d$ for $\bar{h} \in \Sigma^*$ and some fixed set of sequences $\bar{q}_i \in \Sigma^*$. If $\dot{\omega}_{\bar{h}}$ is a sufficient statistic for any history $\bar{h} \in \Sigma^*$, i.e., for every $\bar{x} \in \Sigma^*$ there is a function $m_{\bar{x}} : \mathbb{R}^d \rightarrow [0, 1]$ such that $f(\bar{x} | \bar{h}) = m_{\bar{x}}(\dot{\omega}_{\bar{h}})$ for all $\bar{h} \in \Sigma^*$, then the sequences $\{\bar{q}_1, \dots, \bar{q}_d\}$ are called *core tests*, which together with the initial state $\dot{\omega}_\varepsilon$ and projection functions $m_{\bar{x}}$ form a d -dimensional predictive state representation (PSR) for f . If the projection functions are linear functionals (i.e., just row vectors in $\mathbb{R}^d$), then the PSR is called *linear*.

Note that PSRs share the notion of “state” with OOMs in that the state consists of the information required to predict the future, but PSRs additionally require the entries of the state vectors $\dot{\omega}_{\bar{h}}$ to be “predictions” $f(\bar{q}_i | \bar{h})$ for the core tests $\bar{q}_i$. Such states are therefore called *predictive states*.

4.3 Predictive State Representations and Controlled Processes

We will only consider linear PSRs for controlled processes here, and show that these are essentially SSs for controlled processes (i.e., IO-OOMs) that are additionally interpretable with respect to singleton sets (core tests). Note that there has been some confusion about the precise relationship between PSRs and IO-OOMs, which we address in Sections 4.3.2 and 4.3.3 below.

Proposition 32. *Let a d -dimensional linear PSR consisting of core tests $\bar{q}_i$, projection functions $m_{\bar{x}}$ and an initial state $\dot{\omega}_\varepsilon$ for a controlled process f be given. Then an equivalent SS $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ is obtained by setting*

$$\omega_\varepsilon = \dot{\omega}_\varepsilon, \quad \tau_z = [(m_{z\bar{q}_1})^\top, \dots, (m_{z\bar{q}_d})^\top]^\top \quad \text{and} \quad \sigma = \sum_{o \in \Sigma_O} m_{ao} \text{ for any } a \in \Sigma_I.$$

Furthermore, $\mathcal{M}$ will be interpretable w.r.t. the sets $\{\bar{q}_i\}$.

Proof. First note that $\sigma \dot{\omega}_{\bar{x}} = \sum_{o \in \Sigma_O} m_{ao} \dot{\omega}_{\bar{x}} = \sum_{o \in \Sigma_O} f(ao | \bar{x}) = 1$ for all $\bar{x} \in \Sigma^*$ such that $f(\bar{x}) \neq 0$ because f is a controlled process. Next, we prove that (*) $\omega_{\bar{x}} = f(\bar{x}) \dot{\omega}_{\bar{x}}$ and (**) $f_{\mathcal{M}}(\bar{x}) = f(\bar{x})$ by induction on the length l of $\bar{x}$:

- For $l = 0$ we have $\omega_\varepsilon = f(\varepsilon) \dot{\omega}_\varepsilon$ and $f_{\mathcal{M}}(\varepsilon) = \sigma \omega_\varepsilon = \sigma \dot{\omega}_\varepsilon = 1 = f(\varepsilon)$.
- Assume (*) and (**) are true for all $\bar{x} \in \Sigma^l$. Let $\bar{x}z \in \Sigma^{l+1}$. Then (*) $\omega_{\bar{x}z} = \tau_z \omega_{\bar{x}} = \tau_z \dot{\omega}_{\bar{x}} f(\bar{x}) = [f(z\bar{q}_i | \bar{x})]_{i \in \{1, \dots, d\}}^\top f(\bar{x}) = [f(\bar{q}_i | \bar{x}z)]_{i \in \{1, \dots, d\}}^\top f(\bar{x}z) = \dot{\omega}_{\bar{x}z} f(\bar{x}z)$, and (**) $f_{\mathcal{M}}(\bar{x}z) = \sigma \omega_{\bar{x}z} = \sigma \dot{\omega}_{\bar{x}z} f(\bar{x}z) = f(\bar{x}z)$.

Note that property (*) says that $\omega_{\bar{x}} = f(\bar{x}) \dot{\omega}_{\bar{x}} = [f(\bar{x}\bar{q}_1), \dots, f(\bar{x}\bar{q}_d)]^\top$ for all $\bar{x}$, i.e., that $\mathcal{M}$ is interpretable w.r.t. the sets $\{\bar{q}_i\}$. ■

Proposition 33. *Conversely, let $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be a SS for a controlled process f . Then an equivalent PSR is obtained by making the SS interpretable with respect to singleton sets $\{\bar{y}_i\}$ for appropriate sequences $\bar{y}_i \in \Sigma^*$ (e.g., using Algorithm 3). We can then use these as core tests for the PSR and set $m_{\bar{x}} = \sigma \tau_{\bar{x}}$ for all $\bar{x} \in \Sigma^*$.*

Proof. We assume that the SS has been made interpretable w.r.t. the sequences $\bar{y}_1, \dots, \bar{y}_d$. This means that the normalized states $\dot{\omega}_{\bar{h}} = \omega_{\bar{h}} / \sigma \omega_{\bar{h}}$ have the form $\dot{\omega}_{\bar{h}} = [f(\bar{y}_1 | \bar{h}), \dots, f(\bar{y}_d | \bar{h})]^\top$. Furthermore, for all histories $\bar{h} \in \Sigma^*$ we have $m_{\bar{x}} \dot{\omega}_{\bar{h}} = \sigma \tau_{\bar{x}} \dot{\omega}_{\bar{h}} = \sigma \tau_{\bar{x}} \tau_{\bar{h}} \omega_\varepsilon / \sigma \tau_{\bar{h}} \omega_\varepsilon = f(\bar{x} | \bar{h})$, as desired. ■

Corollary 34. *A linear PSR is specified by the parameters $(\{m_{ao}\}_{ao \in \Sigma}, \{M_{ao}\}_{ao \in \Sigma}, \dot{\omega}_\varepsilon)$ for $\Sigma = \Sigma_I \times \Sigma_O$, where $M_{ao} = [(m_{ao}\bar{q}_1)^\top, \dots, (m_{ao}\bar{q}_d)^\top]$, and defines a controlled process via*

$$f(a_1 o_1 \dots a_n o_n) = \dot{\omega}_\varepsilon^\top M_{a_1 o_1} \dots M_{a_{n-1} o_{n-1}} m_{a_n o_n}^\top.$$

4. VERSIONS OF SEQUENTIAL SYSTEMS

The representation from Corollary 34 is the usual way of specifying a PSR. Transformed PSRs (TPSRs) are just PSRs that model controlled processes in the form of Corollary 34 without any further requirements (i.e., without the requirement that the states need to be interpretable). These are readily converted to SSs by setting $\omega_\varepsilon = \dot{\omega}_\varepsilon$, $\tau_{ao} = M_{ao}^\top$, and $\sigma = \sum_{o \in \Sigma_O} m_{ao}$ for any $a \in \Sigma_I$, since $\sigma \tau_{\bar{x}} \omega_\varepsilon = \sum_{o \in \Sigma_O} \dot{\omega}_\varepsilon^\top M_{a_1 o_1} \cdots M_{a_n o_n} m_{ao}^\top = \sum_{o \in \Sigma_O} f(\bar{x}ao) = f(\bar{x})$ for all $\bar{x} = a_1 o_1 \dots a_n o_n \in \Sigma^*$ by the property (ii) of controlled processes. Note that this conversion may therefore not give equivalent models if the (T)PSR does not model a controlled process.

4.3.1 Relation to Partially Observable Markov Decision Processes

Finally, we note how to convert POMDPs into SSs (which can then be further converted to PSRs by making the SS interpretable, as described above). A POMDP with d states $Q = \{1, \dots, d\}$ for a controlled process with input alphabet Σ_I and output alphabet Σ_O consists of an initial belief state $b \in \mathbb{R}^d$ whose i -th element is the probability of the model starting in state i , a state transition matrix $T_a \in \mathbb{R}^{d \times d}$ for each action $a \in \Sigma_I$ such that the i, j -th entry of T_a is the probability of transitioning from state i to state j if action a is taken, and a vector $E_{ao} \in \mathbb{R}^d$ for each action-observation pair $ao \in \Sigma_I \times \Sigma_O$ whose i -th entry is the probability of observing o after arriving in state i by taking action a [Kaelbling et al., 1998].

We can summarize the belief-state update procedure for the POMDP concisely by stating that a POMDP models a controlled stochastic process f via the equation

$$f(a_1 o_1 \dots a_n o_n) = \mathbb{1}_d^\top \cdot \mathbf{D}(E_{a_n o_n}) T_{a_n}^\top \cdots \mathbf{D}(E_{a_1 o_1}) T_{a_1}^\top \cdot b.$$

Clearly, setting $\sigma = \mathbb{1}_d^\top$, $\tau_{ao} = \mathbf{D}(E_{ao}) T_a^\top$ and $\omega_\varepsilon = b$ yields an equivalent SS.

4.3.2 IO-OOMs, Interpretable IO-OOMs, PSRs, and TPSRs

We have shown above that IO-OOMs, PSRs, and TPSRs are equivalent models in the sense that they model the same class of controlled processes and that they can be readily converted into one another. Furthermore, TPSRs are essentially IO-OOMs (except that the evaluation functional σ is replaced by the set $\{m_{ao}\}$ of evaluation functionals), while PSRs are TPSRs (and therefore essentially IO-OOMs) with predictive states, which corresponds to IO-OOMs being interpretable w.r.t. singleton sets (core tests). This is summarized in Table 1.

4.3 Predictive State Representations and Controlled Processes

SSs for controlled processes with...	single evaluation functional σ	set of evaluation functionals $\{m_{ao}\}$
abstract, uninterpretable states	IO-OOMs	TPSRs
predictive states	IO-OOMs that are interpretable w.r.t. singleton sets	PSRs

Table 1: The differences between IO-OOMs, PSRs, and TPSRs

Note that we have written “IO-OOMs that are interpretable w.r.t. singleton sets” instead of simply “interpretable IO-OOMs” for a reason. This is because interpretability was originally defined for IO-OOMs in a more restrictive way (see Section 4.3.3 below). It has been shown that not every IO-OOM has an equivalent “interpretable IO-OOM” (in the original sense) [Singh et al., 2004], i.e., that “interpretable IO-OOMs” are less general than IO-OOMs and PSRs. At the same time, it was believed that some notion of interpretability would be crucial for the learnability of such models, which is however not the case, as we shall see in Chapter 5. Together, this has led to the false impression that PSRs are more general than IO-OOMs.

As the original notion of interpretability for IO-OOMs has turned out to be overly restrictive, we propose to employ the notion of interpretability that we have introduced here for SSs as the “correct” notion for IO-OOMs and consider the original notion as deprecated.

4.3.3 Historical Remarks

The same remarks that we have made above in Section 4.2.3 for OOMs also apply to IO-OOMs. Namely, IO-OOMs were originally required to satisfy (ii)': $\forall a \in \Sigma_I : \sigma \sum_{o \in \Sigma_O} \tau_{ao} = \sigma$ instead of the the property (ii) for a controlled process. This is equivalent for minimal models but slightly more restrictive in general. However, as every SS can be minimized, this has no practical consequences.

Furthermore, IO-OOMs were originally typically required to satisfy $\sigma = \mathbb{1}_d^\top$, which is again merely a matter of normalization. However, it is only possible for an IO-OOM that satisfies $\sigma = \mathbb{1}_d^\top$ to be interpretable with respect to the sets $Y_1, \dots, Y_d$, if $1 = \sigma \omega_\varepsilon = \mathbb{1}_d^\top [f_{\mathcal{M}}(Y_k)]_{k \in \{1, \dots, d\}}^\top = \sum_{k=1}^d \sum_{\bar{y} \in Y_k} f(\bar{y})$. It turns out that this can be assured by requiring the sets Y_k to partition $\{a_1 \dots a_l\} \times \Sigma_O^l$ for some l and a fixed

4. VERSIONS OF SEQUENTIAL SYSTEMS

sequence $a_1 \dots a_l$ of inputs called a *characterization frame*. This restriction on the choice of sets Y_k therefore became part of the original definition of interpretability for IO-OOMs.

Unlike the case for OOMs, the resulting original notion of interpretability for IO-OOMs has turned out to be a severe limitation [Singh et al., 2004]. However, one may use the more general notion of interpretability given in Definition 16 for IO-OOMs instead, if one is willing to drop the (unnecessary) normalization requirement $\sigma = \mathbb{1}_d^\top$.

4.4 Extensions

In this section we have presented SMAs, OOMs, and PSRs as versions of linear sequential systems – or more generally weighted finite automata – that model probabilistic languages, stochastic processes, and controlled processes, respectively, as is summarized in Figure 1. For completeness, we wish to briefly mention some extensions of these basic model types that have been studied but which are beyond the scope of this paper.

First of all, various non-linear SSs exist. For instance, several versions of *quantum finite automata* have been studied [Kondacs and Watrous, 1997; Moore and Crutchfield, 2000]. One form are SSs $(\sigma, \{\tau_x\}, \omega_\varepsilon \in \mathbb{C}P^d)$ where the operators τ_x are unitary and $\sigma(\tau_{\bar{x}}\omega_\varepsilon) = \|\pi\tau_{\bar{x}}\omega_\varepsilon\|^2$ for some projection π and the Fubini-Study metric $\|\cdot\|$ [Moore and Crutchfield, 2000]. A similar type of OOMs exist which are called *norm-OOMs*. These are SSs $(\sigma, \{\tau_x\}, \omega_\varepsilon \in \mathbb{R}^d)$ such that $\sum_{x \in \Sigma} \tau_x^\top \tau_x = \mathbf{I}$ and $\sigma(\tau_{\bar{x}}\omega_\varepsilon) = \|\tau_x\omega_\varepsilon\|^2$. Such norm-OOMs describe stochastic processes and can always be converted into an equivalent OOM [Zhao and Jaeger, 2010]. Recently, *quadratic weighted automata* have been proposed by Bailly [2011], where a SS $\mathcal{M}$ is learned for $\sqrt{f}$ and a product SS $\mathcal{M} \otimes \mathcal{M}$ is constructed that satisfies $f_{\mathcal{M} \otimes \mathcal{M}} = f_{\mathcal{M}}^2 \approx f$. All of these approaches avoid the “negative probabilities problem”, where the estimated model $f_{\mathcal{M}}$ may violate the requirement $f_{\mathcal{M}} \geq 0$. Non-linear versions of PSRs have also been investigated, which have been shown in some cases to yield representations for deterministic dynamical systems that are exponentially smaller than a minimal OOM representation [Rudary and Singh, 2003].

Furthermore, OOMs and PSRs are models for discrete-valued stochastic (controlled) processes. Many real-world processes of interest are, however, continuous-valued. A continuous version of OOMs exists that extends semi-continuous HMMs [Jaeger, 2000a], and WFST have been similarly extended to allow for continuous inputs [Recasens and Quattoni, 2013]. An alternative approach to dealing with continuous values is to use so-called “binless” OOMs [Wu and No  , 2016]. Multivariate continuous inputs and outputs

are handled using features of observations by reduced-rank HMMs [Siddiqi et al., 2010]. So called predictive linear Gaussian models (PLGs), which are based on PSRs, closely resemble linear dynamical system models [Rudary et al., 2005; Wingate and Singh, 2006a,b; Rudary and Singh, 2006, 2008] and are further generalized by exponential family PSRs [Wingate and Singh, 2008b,a]. A generalization of OOMs using Hilbert space embeddings was introduced by Song et al. [2010]. This has been further refined and extended to include features and can now be employed – among other things – for controlled processes and for planning in reinforcement learning tasks [Boots and Gordon, 2010; Boots et al., 2010, 2013].

4. VERSIONS OF SEQUENTIAL SYSTEMS

5

Learning Sequential Systems

In this chapter we present a general approach to learning SSs from data. We show how several of the learning algorithms that have been proposed for SMA, OOMs, and PSRs can be understood in this framework, and that in fact many of the proposed learning algorithms are closely related.

We begin by establishing a result that lies at the heart of the learning algorithms, which was formulated by Kretzschmar [2003] for the case of OOMs. Assuming a function $f_{\mathcal{M}}$ can be described by some minimal SS $\mathcal{M}$, it allows us to reconstruct an equivalent SS $\mathcal{M}'$ from data given in the form of finitely many function values of $f_{\mathcal{M}}$ – as long as these are given exactly and we know the rank d of the underlying model $\mathcal{M}$. We will therefore refer to the Equations (2) as the *learning equations*.

Proposition 35. *For a minimal d -dimensional SS $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$, consider finite sets of states $\{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in X\}$ and co-states $\{(\sigma\tau_{\bar{y}})^\top \mid \bar{y} \in Y\}$ that span the state space W and the co-state space $\tilde{W}$, respectively. Define the matrices $F_{Y,X} = [f_{\mathcal{M}}(\bar{x}\bar{y})]_{\bar{y} \in Y, \bar{x} \in X}$ and $F_{Z,Y,X} = [f_{\mathcal{M}}(\bar{x}z\bar{y})]_{\bar{y} \in Y, \bar{x} \in X}$. Furthermore, define the vector $F_Y^\top = [f_{\mathcal{M}}(\bar{y})]_{\bar{y} \in Y}^\top$ and row vector $F_X = [f_{\mathcal{M}}(\bar{x})]_{\bar{x} \in X}$. Let $C \in \mathbb{R}^{d \times |Y|}$ and $Q \in \mathbb{R}^{|X| \times d}$ be rank- d matrices such that $CF_{Y,X}Q$ is invertible, which exist, since $F_{Y,X}$ has rank d . Then the SS $\mathcal{M}' = (\sigma', \{\tau'_z\}, \omega'_\varepsilon)$ defined as follows is equivalent to $\mathcal{M}$:*

$$\begin{aligned}\sigma' &= F_X Q (CF_{Y,X} Q)^{-1}, \\ \tau'_z &= CF_{Z,Y,X} Q (CF_{Y,X} Q)^{-1}, \\ \omega'_\varepsilon &= CF_Y^\top.\end{aligned}\tag{2}$$

Furthermore, we have $CF_{Y,X} = [\omega'_x]_{\bar{x} \in X}$ and $CF_{Z,Y,X} = [\omega'_{xz}]_{\bar{x} \in X}$, where $\omega'_x = \tau'_x \omega'_\varepsilon$ are states of the SS $\mathcal{M}'$.

5. LEARNING SEQUENTIAL SYSTEMS

Proof. By Corollary 10 we have $\text{rank}(\mathcal{M}) = d$ and $W = \tilde{W} = \mathbb{R}^d$, hence $\Pi = [(\sigma\tau_{\bar{y}})^\top]_{\bar{y} \in Y}^\top$ and $\Phi = [\tau_{\bar{x}}\omega_\varepsilon]_{\bar{x} \in X}$ have full rank d . We have $F_{Y,X} = \Pi\Phi$, $F_{zY,X} = \Pi\tau_z\Phi$, $F_Y^\top = \Pi\omega_\varepsilon$, and $F_X = \sigma\Phi$, so indeed $\text{rank}(F_{Y,X}) = d$. We can then simply calculate $\tau'_z = C\Pi\tau_z\Phi Q(C\Pi\Phi Q)^{-1} = C\Pi\tau_z(C\Pi)^{-1}$, as well as $\omega'_\varepsilon = C\Pi\omega_\varepsilon$ and $\sigma' = \sigma\Phi Q(C\Pi\Phi Q)^{-1} = \sigma(C\Pi)^{-1}$, showing that $\mathcal{M}' = \rho\mathcal{M}\rho^{-1}$ for the non-singular transformation $\rho = C\Pi$. Furthermore, $CF_{Y,X} = C\Pi\Phi = \rho\Phi = [\rho\tau_{\bar{x}}\omega_\varepsilon]_{\bar{x} \in X} = [\tau'_{\bar{x}}\omega'_\varepsilon]_{\bar{x} \in X}$ and analogously for $CF_{zY,X}$. ■

The matrices C and Q that appear in the learning Equations (2) are indeed arbitrary (provided that $CF_{Y,X}Q$ has the correct dimension d and full rank), as long as the function values $f_{\mathcal{M}}(\bar{x})$ are given exactly. However, if one only has access to estimates $\hat{f}(\bar{x})$, then the selection of C and Q plays a crucial role in obtaining good model estimates, as will be further discussed in Section 5.4.

Furthermore, note that we generally do not know a priori which sets X and Y of words to consider such that $\{\tau_{\bar{x}}\omega_\varepsilon \mid \bar{x} \in X\}$ and $\{(\sigma\tau_{\bar{y}})^\top \mid \bar{y} \in Y\}$ span the state and co-state spaces W and $\tilde{W}$ of $\mathcal{M}$. Proposition 8 guarantees that it suffices to consider all words of length at most d , but the rank d of $\mathcal{M}$ is generally unknown as well. Selecting appropriate sets X and Y of words and an appropriate model dimension d are, therefore, crucial and non-trivial steps in learning models from data.

We can turn the above Proposition 35 into a generic learning procedure for SSs:

Algorithm 4: General procedure for learning a SS from data

- 1 Obtain estimates $\hat{f}(\bar{x})$ of the function values $f(\bar{x})$ for words $\bar{x} \in \Sigma^*$.
 - 2 Choose finite sets $X, Y \subset \Sigma^*$, which we call sets of *indicative* and *characteristic* words, respectively. Then assemble the estimates $\hat{f}(\bar{x})$ into estimates of the matrices $\hat{F}_{Y,X}$, $\hat{F}_{zY,X}$, $\hat{F}_Y^\top$ and $\hat{F}_X$.
 - 3 Find a reasonable target dimension d for the model to be learned.
 - 4 Choose $C \in \mathbb{R}^{d \times |Y|}$ and $Q \in \mathbb{R}^{|X| \times d}$ called the *characterizer* and *indicator* such that $C\hat{F}_{Y,X}Q$ is invertible.
 - 5 Apply the learning Equations (2) to obtain a model estimate $\hat{\mathcal{M}}$.
-

At this point we should clarify what is meant here by learning a model from data. For general MA the goal is often to reconstruct an automaton from as few *membership queries* – obtaining the value $f(\bar{x})$ for some $\bar{x} \in \Sigma^*$ – and *equivalence queries* – proposing a function h and receiving a *counterexample* $\bar{x}$ such that $h(\bar{x}) \neq f(\bar{x})$ if $h \neq f$ – as possible. This is an extended version of the exact learning model of Angluin [1987].

However, in the case of SMA, OOMs and PSRs, the external function represents a distribution. Therefore, in these cases it is usual to assume that we observe samples from this distribution and wish to estimate model parameters from the given samples such that the estimated model best describes the underlying distribution – “best” in a sense that depends on the context and the approach taken by a specific learning algorithm.

Note that learning algorithms based on the general learning procedure from Algorithm 4 typically do not attempt to maximize the model likelihood for the available training data. However, it is possible to use gradient descent on the SS model parameters to optimize this likelihood [Jiang et al., 2016]. While this is impractical as a SS learning algorithm in its own right, it can be used to further optimize a SS that is learned for instance using a procedure as in Algorithm 4.

We should also mention a common problem when estimating SMAs, OOMs, and PSRs from sample data. Namely, even if the function $f_{\mathcal{M}}$ in question can be described by a SMA, OOM, or PSR model $\mathcal{M}$, the learned model $\hat{\mathcal{M}}$ will only be an approximation to $\mathcal{M}$ and will describe a function $f_{\hat{\mathcal{M}}}$ that may not satisfy the properties of a probabilistic language, stochastic process, or controlled process, respectively, i.e., the learned model $\hat{\mathcal{M}}$ may *not* be a SMA, OOM, or PSR. What typically happens is that the learned model $\hat{\mathcal{M}}$ will predict “negative probabilities” for certain sequences $\bar{x}$. Moreover, it is an undecidable problem whether a given SS $\hat{\mathcal{M}}$ satisfies $f_{\hat{\mathcal{M}}} \geq 0$, and therefore, whether it is a SMA – a result that carries over to OOMs and PSRs as well [Wiewiora, 2008]. In practice, there are several ways to deal with this “negative probabilities problem”:

First of all, one can resort to alternative models as described in Section 4.4 that preclude the problem by design. For the particular case of quadratic weighted automata the learning procedure presented here still applies [Bailly, 2011], but in general one will need alternative learning algorithms.

Secondly, one may attempt to learn a restricted class of SSs such as PFA, HMMs, or POMDPs by enforcing additional constraints on the parameters of the SS. This can be achieved by employing a non-negative version of the spectral learning method we present in Section 5.4.2 [Glaude et al., 2015]. Alternatively, one may learn a SS model first and then apply an additional conversion step [Hsu et al., 2009; Anandkumar et al., 2012], which, however, may fail. A variant of this last approach is to use a generalization of the spectral learning method to learn the parameters of a latent-variables model such as a HMM directly [Anandkumar et al., 2014], but this approach can still result in invalid models. Nevertheless, it is possible to use such a learned SS model as an

5. LEARNING SEQUENTIAL SYSTEMS

initialization to an iterative exterior point method for optimizing the desired latent-variables model [Shaban et al., 2015].

Finally, one may work with such an “invalid” SS model by employing a simple and effective heuristic as suggested by Jaeger et al. [2006b] to normalize all model predictions to fall into the desired range. We describe a slightly generalized version of this heuristic in Section 5.6.

In the following, we will discuss the steps of the learning procedure in more detail.

5.1 Obtaining Estimates $\hat{f}(\bar{x})$

Here we explain how to obtain estimates $\hat{f}(\bar{x})$ for words $\bar{x} \in \Sigma^*$ from the available training data, which depends on the type of data we are dealing with. Recall that in the context of SMA, the functions we are considering are distributions on words, while in the context of OOMs and PSRs they represent stochastic processes and controlled processes, respectively. The following Remarks 36 and 37 summarize how to obtain these estimates in the different scenarios of probabilistic languages, stochastic processes, and controlled processes, respectively.

Remark 36. *Let $f : \Sigma^* \rightarrow [0, 1]$ be a distribution on Σ^* , and let $S = \{\bar{s}_1, \bar{s}_2, \dots, \bar{s}_N\}$ be a collection of N samples from f . Then $\hat{f}(\bar{x}) = \frac{\#(\bar{x})}{N}$, where $\#(\bar{x})$ denotes the number of occurrences of $\bar{x}$ in S , is an unbiased and consistent estimator for $f(\bar{x})$.*

In the case of stochastic processes, one typically observes a finite number of values from one, or possibly several, samples of the stochastic process. Note that such a sample $\bar{s}$ is technically a word $s_1 s_2 \dots$ of infinite length. We will therefore use the terms *initial realization* (of length n) to refer to the first n values $s_1 s_2 \dots s_n$ of a sample $\bar{s}$, and *partial realization* (of length n) to refer to n consecutive values $s_t s_{t+1} \dots s_{t+n-1}$ of a sample $\bar{s}$ starting from some unknown time t . It is possible to obtain consistent estimates $\hat{f}(\bar{x})$ even from a single initial realization if the stochastic process is stationary and ergodic¹ by using time-averages instead of space-averages as estimates. The same idea is commonly used for the case of controlled processes as well and called the *suffix-history* method in the PSR community. For simplicity, we will consider only data consisting of a single initial realization, but note that it is straight forward to generalize this to data consisting of several initial realizations. Recently, this has been generalized to

¹A stationary ergodic process is a stochastic process where the statistical properties do not change with time (stationarity) and where these can be consistently estimated as time-averages from a single sample path (ergodicity). For details, see for example the textbook by Gray [1988].

the practically very relevant case of data consisting of several partial realizations – so called “non-equilibrium” data [Wu and Noé, 2016].

We will now turn immediately to the more general case of controlled processes, since one may view any stochastic process as a special case of a controlled process with an input alphabet Σ_I of size one, and the estimators presented below in Remark 37 therefore apply to stochastic processes as well.

It is important to have a good understanding of the meaning of the value $f(\bar{x})$ when f is a controlled process and $\bar{x} = a_1 o_1 \dots a_n o_n \in (\Sigma_I \times \Sigma_O)^n$ is some input-output sequence. Intuitively, this is the probability of the system output $o_1 \dots o_n$ conditioned on the system input $a_1 \dots a_n$. However, in general $f(a_1 o_1 \dots a_n o_n)$ is *not* equal to $\mathbb{P}(o_1 \dots o_n | a_1 \dots a_n) = \frac{\mathbb{P}(a_1 o_1 \dots a_n o_n)}{\mathbb{P}(a_1 \dots a_n)}$ [Bowling et al., 2006]. To clarify this, consider the stochastic process that is specified by the controlled process f *together with some system input specification*. This stochastic process is governed by probabilities of the form

$$\mathbb{P}(a_1 o_1 \dots a_n o_n) = \underbrace{\prod_{k=1}^n \mathbb{P}(o_k | a_1 o_1 \dots a_k)}_{f(\bar{x})} \cdot \underbrace{\prod_{k=1}^n \mathbb{P}(a_k | a_1 o_1 \dots a_{k-1} o_{k-1})}_{\pi(\bar{x})}. \quad (3)$$

The second factor in the equation models the system input and is sometimes called the *input policy* π , while the first factor models the system output and is just the controlled process f . Therefore, for $\bar{x} = a_1 o_1 \dots a_n o_n$,

$$f(\bar{x}) = \prod_{k=1}^n p_k(\bar{x}) = \frac{\mathbb{P}(\bar{x})}{\pi(\bar{x})}, \quad \text{where} \quad (4)$$

$$p_k(\bar{x}) = \mathbb{P}(o_k | a_1 o_1 \dots a_k).$$

Note that for the special case of a *blind* input policy π that does not depend on the observed output, i.e., that satisfies $\mathbb{P}(a_k | a_1 o_1 \dots a_{k-1} o_{k-1}) = \mathbb{P}(a_k | a_1 \dots a_{k-1})$ for all $\bar{x}$, we do in fact have $\pi(\bar{x}) = \mathbb{P}(a_1 \dots a_n)$ and hence $f(\bar{x}) = \mathbb{P}(o_1 \dots o_n | a_1 \dots a_n)$.

Equation (4) motivates the following estimators.

Remark 37. Let $f : \Sigma^* \rightarrow [0, 1]$ be a controlled process and $\pi > 0$ be some input policy, such that the resulting stochastic process is stationary and ergodic, and let $\bar{s} = a_1 o_1 \dots a_N o_N$ be an initial realization of the resulting stochastic process. Then

$$\hat{f}(\bar{x}) = \prod_{k=1}^{|\bar{x}|} \hat{p}_k(\bar{x}) \quad \text{with} \quad \hat{p}_k(\bar{x}) = \begin{cases} \frac{\#(a_1 o_1 \dots a_k o_k)}{\#(a_1 o_1 \dots a_k)} & \text{if } \#(a_1 o_1 \dots a_k) > 0, \\ \frac{1}{|\Sigma_O|} & \text{otherwise} \end{cases}$$

5. LEARNING SEQUENTIAL SYSTEMS

is a consistent estimator for $f(\bar{x})$. If the input policy π is known, then

$$\hat{f}_\pi(\bar{x}) = \frac{\#(\bar{x})}{N - |\bar{x}| + 1} \cdot \frac{1}{\pi(\bar{x})}$$

is another consistent and unbiased estimator for $f(\bar{x})$, where again $\#(\bar{x})$ denotes the number of occurrences of $\bar{x}$ in the sequence $\bar{s}$.

These estimators also apply to stochastic processes, where $\Sigma = \{1\} \times \Sigma_O$ and $\pi = 1$.

To understand the properties of the estimator $\hat{f}(\bar{x})$ from Remark 37, we consider instead a very similar estimator $\hat{f}'(\bar{x})$, which uses the correct but unknown value $p_k(\bar{x})$ in case the word $a_1 o_1 \dots a_k$ does not occur.

$$\hat{f}'(\bar{x}) = \prod_{k=1}^{|\bar{x}|} \hat{p}'_k(\bar{x}) \quad \text{with} \quad \hat{p}'_k(\bar{x}) = \begin{cases} \frac{\#(a_1 o_1 \dots a_k o_k)}{\#(a_1 o_1 \dots a_k)} & \text{if } \#(a_1 o_1 \dots a_k) > 0, \\ p_k(\bar{x}) & \text{otherwise.} \end{cases}$$

The following Lemma 38 is due to Bowling et al. [2006], and we include their proof for completeness¹.

Lemma 38. *The estimators $\hat{p}'_k(\bar{x})$ for $p_k(\bar{x})$ defined above are unbiased and uncorrelated. The estimator $\hat{f}'(\bar{x})$ for $f(\bar{x})$ is therefore also unbiased.*

Proof. (Bowling et al. [2006]) Let I_n be any subset of $\{1, 2, \dots, n\}$ containing n . Then

$$\begin{aligned} \mathbb{E} \left[\prod_{k \in I_n} \hat{p}'_k(\bar{x}) \right] &\stackrel{(1)}{=} \mathbb{E} \left[\mathbb{E} \left[\prod_{k \in I_n} \hat{p}'_k(\bar{x}) \mid \begin{array}{l} \#(a_1 o_1 \dots a_n), \\ \forall k \in I_n \setminus \{n\} : \#(a_1 o_1 \dots a_k), \\ \forall k \in I_n \setminus \{n\} : \#(a_1 o_1 \dots a_k o_k) \end{array} \right] \right] \\ &\stackrel{(2)}{=} \mathbb{E} \left[\left(\prod_{k \in I_n \setminus \{n\}} \hat{p}'_k(\bar{x}) \right) \mathbb{E}[\hat{p}'_n(\bar{x}) \mid \#(a_1 o_1 \dots a_n)] \right] \\ &\stackrel{(3)}{=} \mathbb{E} \left[\left(\prod_{k \in I_n \setminus \{n\}} \hat{p}'_k(\bar{x}) \right) \mathbb{P}(o_n \mid a_1 o_1 \dots a_n) \right] \\ &\stackrel{(4)}{=} \prod_{k \in I_n} \mathbb{P}(o_k \mid a_1 o_1 \dots a_k), \end{aligned}$$

where (1) used the law of total expectation, (2) pulled out from the inner expectation the

¹In fact, Bowling et al. omitted the special treatment for $\#(a_1 o_1 \dots a_k) = 0$ altogether and ignored the issue that the resulting estimator is then not always defined. Also, they do not explicitly state the uncorrelatedness of the $\hat{p}'_k(\bar{x})$.

terms that depend only on the conditioning random variables, (3) uses the stationarity of the underlying process, and (4) follows by induction on n starting from the trivial statement that $\mathbb{E}[1] = 1$ for $n = 0$. $\blacksquare$

In practice, we do not know $\mathbb{P}(o_k | a_1 o_1 \dots a_k)$, so we must resort to the estimator $\hat{f}(\bar{x})$ instead. Furthermore, note that if $\pi(\bar{x}) = 0$ for some $\bar{x}$, then the input sequence corresponding to $\bar{x}$ cannot occur under the policy π , and therefore no information can be gained about the system output $f(\bar{x})$ for these cases, which means that the estimator $\hat{f}(\bar{x})$ may not be consistent. The common solution is to simply require $\pi > 0$.

None of the above estimates exploits the rich structure of the matrix F . If required, some of the convex constraints that the matrix F must satisfy can be ensured by applying an additional normalization step to the estimated matrix $\hat{F}$, as done by McCracken and Bowling [2006]. These convex constraints – including a convex relaxation of the rank constraint – may also be used to infer missing values if some entries $\hat{f}(\bar{x})$ cannot be obtained directly, which becomes relevant in the context of learning more general (e.g., non-stochastic) weighted automata [Balle and Mohri, 2012] or to infer sequence alignment when learning WFST from unaligned input-output sequences [Bailly et al., 2013].

5.1.1 Variance Estimates

In addition to the estimates $\hat{f}(\bar{x})$ themselves, we can also estimate their variance $\mathbb{V}[\hat{f}(\bar{x})]$. Later, in Section 5.4.3, we will make use of this additional information by introducing weights into the learning algorithm that measure the *precision* of the estimates $\hat{f}(\bar{x})$ and are given by $\hat{\mathbb{V}}[\hat{f}(\bar{x})]^{-1}$. For this, it will be crucial that the variance estimates are never too close to zero. On the other hand, we will also use the variance information to estimate a suitable model dimension in Section 5.3.1, for which it turns out that we require reasonably unbiased variance estimates that are zero if $\hat{f}(\bar{x})$ is zero.

First, for a binomially distributed random variable $X \sim \text{Bin}(n, p)$, we consider the unbiased and consistent estimator $\hat{p} = \frac{x}{n}$ of the parameter p . The variance of $\hat{p}$ is $\frac{p(1-p)}{n}$, and an unbiased estimator for the variance is the *sample variance* $\hat{\mathbb{V}}[\hat{p}] = \frac{\hat{p}(1-\hat{p})}{n-1}$. However, this variance estimate may be zero or more generally may severely underestimate the true variance, especially if n is small and/or p is either very small or very large. It is also undefined for $n < 2$. We will address this by a combination of the following approaches.

1. We may wish to enforce a lower bound for the estimated variance, or better, a lower bound that depends on the sample size n , i.e., $\hat{\mathbb{V}}[\hat{p}] \geq \frac{M}{n^2}$ for some constant

5. LEARNING SEQUENTIAL SYSTEMS

M (e.g., $M = 3$). This can be motivated by requiring a minimum number M' of successes and failures of the underlying Bernoulli trial, hence $M' \leq X \leq n - M'$ with $M = M'(1 - \frac{M'}{n})$, where $M \approx M'$ if $M' \ll n$. We then simply *add* the lower bound $\frac{M}{n^2}$ to the variance estimate.

2. We can use Bayesian estimates with a suitably chosen prior distribution for the unknown parameter p . We will use a $\text{Beta}(K, K)$ conjugate prior and compute $\tilde{p} = \frac{x+K}{n+2K}$ as the posterior mean, which can be interpreted as adding $2K$ pseudo-observations. We will generally use $K = 1$, which corresponds to the uniform (Laplace) prior. We then plug this Bayesian estimate $\tilde{p}$ into the formula for the sample variance of $\hat{p}$ (taking into account the additional pseudo-observations) to obtain $\hat{\mathbb{V}}[\hat{p}] = \frac{\tilde{p}(1-\tilde{p})}{n+2K-1}$.
- 2' If we do not want to introduce a bias using a Bayesian prior (i.e., use $K = 0$), we need to specify how to deal with the cases $n = 0$ and $n = 1$. We then set

$$\hat{\mathbb{V}}[\hat{p}] = \begin{cases} 0 & \text{if } n = 1 \text{ and } \hat{p} = 0 \\ 0.25 & \text{otherwise if } n < 2. \end{cases}$$

This endows the variance estimates $\hat{\mathbb{V}}[\hat{f}(\bar{x})]$ obtained via Equation (5) below with the property that $\hat{\mathbb{V}}[\hat{f}(\bar{x})] = 0$ whenever $\hat{f}(\bar{x}) = 0$ (cf. Section 5.3.1).

The above approaches can be combined and applied directly to obtain suitable variance estimates for the estimator from Remark 36 that covers the case of stochastic languages, since in that case the counts $\#(\bar{x})$ are clearly binomially distributed. This is, however, *not* true for the occurrence counts $\#(\bar{x})$ gathered in the case of (controlled) stochastic processes where we resort to time-averages instead of space-averages, which will generally have a larger variance. We will simply ignore this issue and use the same variance estimate for the estimator $\hat{f}_\pi(\bar{x})$ for the case of (controlled) stochastic processes with known policy, even though this may incur considerable error.

Remark 39. Let $\hat{f}(\bar{x}) = \frac{\#(\bar{x})}{n}$ for $n = N \gg 0$ be the estimator from Remark 36 for the case of stochastic languages. Then, for a suitable choice of parameters K and M , we may estimate the variance of $\hat{f}(\bar{x})$ by

$$\begin{aligned} \hat{\mathbb{V}}_{K,M}[\hat{f}(\bar{x})] &= \hat{\mathbb{V}}_{K,M}\left[\frac{\#(\bar{x})}{n}\right] = \frac{\tilde{p}(1-\tilde{p})}{n+2K-1} + \frac{M}{n^2} \\ \text{where } \tilde{p} &= \frac{\#(\bar{x}) + K}{n + 2K - 1}. \end{aligned}$$

Let $\hat{f}_\pi(\bar{x}) = \frac{\#(\bar{x})}{n\pi(\bar{x})}$ for $n = N - |\bar{x}| + 1 \gg 0$ be the estimator from Remark 37 for the case of (controlled) stochastic processes with known policy. Using the definition of $\tilde{p}$ above, we may estimate the variance of $\hat{f}_\pi(\bar{x})$ by

$$\begin{aligned} \hat{\mathbb{V}}_{K,M}[\hat{f}_\pi(\bar{x})] &= \hat{\mathbb{V}}_{K,M}\left[\frac{\#(\bar{x})}{n\pi(\bar{x})}\right] = \frac{1}{\pi(\bar{x})^2} \left(\frac{\tilde{p}(1-\tilde{p})}{n+2K-1} \right) + \frac{M_\pi}{n^2} \\ \text{where } M_\pi &= \min\left\{\max\left\{M, \frac{1}{\pi(\bar{x})^2}\right\}, n^2 M\right\}. \end{aligned}$$

Note that in the latter case we use a slightly modified lower bound $\frac{M_\pi}{n^2}$ that is essentially $\frac{1}{n^2\pi(\bar{x})^2}$, but additionally enforce $\frac{M}{n^2} \leq \frac{M_\pi}{n^2} \leq M$, so that we recover the bound of $\frac{M}{n^2}$ for stochastic processes where $\pi = 1$ and also prevent extremely large variance estimates.

We now turn to the estimator $\hat{f}(\bar{x}) = \hat{p}_1(\bar{x}) \cdots \hat{p}_{|\bar{x}|}(\bar{x})$ from Remark 37 that can be used in general for the case of stochastic processes and controlled processes. To obtain a variance estimate, we need to compute the variance of the product of the random variables $\hat{p}_k(\bar{x})$, which we will evaluate only approximately. We will ignore the subtle difference between the estimators $\hat{p}_k(\bar{x})$ and $\hat{p}'_k(\bar{x})$ and regard the former as unbiased and uncorrelated as well. Moreover, we will assume independence, which means we drop higher-order covariance terms in the computation of the variance of their product. Using the notation $\hat{f}_k(\bar{x}) = \hat{p}_1(\bar{x}) \cdots \hat{p}_k(\bar{x})$ and setting $\hat{f}_0(\bar{x}) = 1$, $\mathbb{V}[1] = 0$, and $\mathbb{E}[1] = 1$, we use the following equation to compute the variance of the product $\hat{f}(\bar{x}) = \hat{p}_1(\bar{x}) \cdots \hat{p}_{|\bar{x}|}(\bar{x})$ recursively.

$$\begin{aligned} \mathbb{V}[\hat{f}_k(\bar{x})] &= \mathbb{V}[\hat{f}_{k-1}(\bar{x})\hat{p}_k(\bar{x})] \\ &= \mathbb{V}[\hat{f}_{k-1}(\bar{x})] (\mathbb{E}[\hat{p}_k(\bar{x})^2] - \mathbb{V}[\hat{p}_k(\bar{x})]) + \mathbb{E}[\hat{f}_{k-1}(\bar{x})^2] \mathbb{V}[\hat{p}_k(\bar{x})] \end{aligned} \tag{5}$$

Substituting $\hat{p}_k(\bar{x})^2$ for $\mathbb{E}[\hat{p}_k(\bar{x})^2]$, $\hat{f}_{k-1}(\bar{x})^2$ for $\mathbb{E}[\hat{f}_{k-1}(\bar{x})^2]$, and an unbiased estimate for $\mathbb{V}[\hat{p}_k(\bar{x})]$ would result in an unbiased variance estimate $\hat{\mathbb{V}}[\hat{f}_k(\bar{x})]$. We will estimate the variance of $\hat{p}_k(\bar{x})$ as if the counts $\#(a_1 o_1 \dots a_k o_k)$ were binomially distributed as $\#(a_1 o_1 \dots a_k o_k) \sim \text{Bin}(\#(a_1 o_1 \dots a_k), p_k(\bar{x}))$, which would be correct if we were using space-averages but is no longer true when using time-averages. Additionally, we will substitute Bayesian estimates using $2K$ pseudo-observations for $\hat{p}_k(\bar{x})$ throughout and also add a term to the final variance estimate to enforce a lower bound controlled by M . This leads to the variance estimation procedure described in the following Remark 40.

5. LEARNING SEQUENTIAL SYSTEMS

Remark 40. Let $\hat{f}(\bar{x}) = \hat{p}_1(\bar{x}) \cdots \hat{p}_{|\bar{x}|}(\bar{x})$ be the estimator from Remark 37 for the case of (controlled) stochastic processes. Then, for a suitable choice of parameters K and M , we may compute an estimate $\hat{\mathbb{V}}_{K,M}[\hat{f}(\bar{x})]$ of the variance of $\hat{f}(\bar{x})$ by the following recursive procedure, where we use the notation $\hat{f}_k = \hat{p}_1(\bar{x}) \cdots \hat{p}_{k-1}(\bar{x})$ and set $\hat{\mathbb{V}}[1] = 0$. All quantities depend on the (fixed) word $\bar{x}$, which we drop for clarity.

$$\begin{aligned}
 \hat{\mathbb{V}}_{K,M}[\hat{f}(\bar{x})] &= \hat{\mathbb{V}}[\hat{f}_{|\bar{x}|}] + \frac{M_{\hat{\pi}}}{N^2}, \\
 \text{where} \quad \hat{\mathbb{V}}[\hat{f}_k] &= \hat{\mathbb{V}}[\hat{f}_{k-1}] \left(\tilde{p}_k^2 - \hat{\mathbb{V}}[\hat{p}_k] \right) + (\tilde{p}_1 \cdots \tilde{p}_{k-1})^2 \hat{\mathbb{V}}[\hat{p}_k] \\
 \text{for} \quad \hat{\mathbb{V}}[\hat{p}_k] &= \begin{cases} \frac{\tilde{p}_k(1-\tilde{p}_k)}{n_k+2K-1} & \text{if } n_k + 2K \geq 2 \\ 0 & \text{if } n_k + 2K = 1 \text{ and } \tilde{p}_k = 0 \\ 0.25 & \text{otherwise} \end{cases} \\
 \text{where} \quad \tilde{p}_k &= \begin{cases} \frac{\#(a_1 o_1 \dots a_k o_k) + K}{n_k + 2K} & \text{if } n_k + 2K > 0 \\ 0.5 & \text{otherwise} \end{cases} \\
 n_k &= \#(a_1 o_1 \dots a_k), \\
 \text{and} \quad M_{\hat{\pi}} &= \min \left\{ \max \left\{ M, \frac{1}{\hat{\pi}(\bar{x})^2} \right\}, N^2 M \right\}, \\
 \text{for} \quad \hat{\pi}(\bar{x}) &= \prod_{k=1}^{|\bar{x}|} \begin{cases} \frac{\#(a_1)}{N} & \text{if } k = 1, \\ \frac{\#(a_1 o_1 \dots a_k)}{\#(a_1 o_1 \dots a_{k-1} o_{k-1})} & \text{if } \#(a_1 o_1 \dots a_{k-1} o_{k-1}) \neq 0, \\ \frac{1}{|\Sigma_I|} & \text{otherwise.} \end{cases}
 \end{aligned}$$

Note that we enforce essentially the same minimum for the variance estimate that we have used above in the case of the estimator $\hat{f}_\pi(\bar{x})$, except that here we use an estimate $\hat{\pi}(\bar{x})$ in place of the actual but unknown value $\pi(\bar{x})$. This is motivated by the relationship between the estimators $\hat{f}_\pi(\bar{x})$ and $\hat{f}(\bar{x})$, namely $\hat{f}_\pi(\bar{x}) \approx \hat{f}(\bar{x}) \frac{\hat{\pi}(\bar{x})}{\pi(\bar{x})}$. We can even say a little more about their relationship. Under the assumption of unbiasedness of $\hat{\pi}(\bar{x})$, we obtain $\mathbb{V}[\hat{f}_\pi(\bar{x})] \approx \text{Cov}[\hat{f}(\bar{x})^2, \frac{\hat{\pi}(\bar{x})^2}{\pi(\bar{x})^2}] + \mathbb{V}[\hat{f}(\bar{x})] \left(\mathbb{V}[\frac{\hat{\pi}(\bar{x})}{\pi(\bar{x})}] + 1 \right) + \mathbb{E}[\hat{f}(\bar{x})]^2 \mathbb{V}[\frac{\hat{\pi}(\bar{x})}{\pi(\bar{x})}]$. We cannot really control the covariance term, but we conjecture it to be small in relation to the other terms and then get $\mathbb{V}[\hat{f}_\pi(\bar{x})] \geq \mathbb{V}[\hat{f}(\bar{x})]$. We conclude that one should generally use the estimator $\hat{f}(\bar{x})$ in favor of the estimator $\hat{f}_\pi(\bar{x})$, even when the input policy is known.

5.2 Choosing Indicative and Characteristic Words

Choosing indicative and characteristic words $X, Y \subset \Sigma^*$ is equivalent to selecting which columns indexed by X and rows indexed by Y of the system matrix F to estimate. Clearly, it is only possible to obtain a correct model for f if X and Y are selected such that $\text{rank}(F) = d = \text{rank}(F_{Y,X})$. We will call such sets of indicative and characteristic words *sufficient*. However, it is unclear how to satisfy this if the true rank is unknown, and this may even be impossible if $\text{rank}(F) = \infty$ – as may often be the case for real-world examples. Estimating the rank and an appropriate dimension for the model will be discussed in the following section.

One approach is to attempt to select minimal sets of indicative and characteristic words such that $\text{rank}(F) = \text{rank}(F_{Y,X})$. Such minimal sufficient sets are called sets of *core histories* and *core tests* in the context of PSRs, and their selection is called the *discovery problem*. This problem is easily solved by Algorithm 1 once a SS model for f is known. For the case where only function values of f are available, an iterative procedure has been proposed [James and Singh, 2004] that, starting with the empty word, adds in each iteration all length-one extensions of previously found core histories and tests but retains only a minimal set needed to span $\hat{F}_{Y,X}$. Since any noisy matrix is typically non-singular, some notion of numerical linear independence is used to decide which words to retain in each step. Unfortunately, there are simple examples of finite rank where this iterative procedure fails to deliver sets of core histories and tests [James and Singh, 2004], i.e., it does not in general solve the discovery problem. A similar algorithm called DEES has been proposed in the context of learning SMA [Denis et al., 2006]. The algorithms for learning MA in the exact learning framework also work by finding minimal sufficient sets of indicative and characteristic words, but there it is assumed that the function f may be queried exactly, and furthermore equivalence queries are employed to find additional core tests and histories [Ohnishi et al., 1994; Bergadano and Varricchio, 1994; Beimel et al., 2000].

For the case where the model dimension d is less than $\text{rank}(F)$, an approach to optimally selecting “minimal” sets X and Y of size d such that $\text{rank}(F_{Y,X}) = d$ was recently proposed [Kulesza et al., 2015b].

It is important to note that there is no requirement to find minimal or even small sets of indicative and characteristic words, i.e., one does not need to solve the discovery problem when learning SS models from data (and once a SS model has been learned, the problem is easily solved by Algorithm 1). In fact, using small such sets means that less of the available training data will enter the model estimation, i.e., the available data

5. LEARNING SEQUENTIAL SYSTEMS

will be under-exploited. It is therefore desirable to use (much) larger sets of indicative and characteristic words than strictly needed.

An approach that is in some sense complementary is to use all words of a given length L . By Proposition 8 one can ensure $\text{rank}(F_{Y,X}) = d$ by choosing $L \geq d$. However, this is highly impractical, since the size of $\hat{F}_{Y,X}$ grows exponentially with L . Also, many of the estimates in $\hat{F}_{Y,X}$ will be based on very few – if any – occurrences in the available training data. Nevertheless, choosing a length $L \ll d$, for instance $L = 1$ or $L = 2$, and utilizing as indicative as well as characteristic words all words of length L or alternatively of length $\leq L$ can give satisfactory results [Zhao et al., 2009a; Hsu et al., 2009; Siddiqi et al., 2010].

It is possible to restrict this selection to words that occur at least with a certain minimum frequency in the available training data [Jaeger et al., 2006b; Thon, 2008] or even just use as indicative and characteristic words all words that occur in the training data, irrespective of the word length [Bailly et al., 2009], although this leads to a very large and sparse matrix $\hat{F}_{Y,X}$. Alternatively, one can select only the words that occur most frequently [Balle et al., 2014]. We should be precise about what we mean by the occurrence frequency of a word in the training data. In the case of (controlled) processes, where the training data consists of one or possibly several long sequences, we count how often a given word occurs as a subsequence in the data. In the case of stochastic languages, however, where the training data consists of a set of words, we count how often a word occurs as a prefix in the training data when considering the word as an indicative word and count how often it occurs as a suffix in the training data when considering the word as a characteristic word. In the case of (controlled) stochastic processes only, a possible refinement employed by Jaeger et al. [2006b] is to disallow indicative words that are suffixes of some other indicative word if they always occur at the same positions in the training data. The reasoning is that such words would just lead to identical columns in the estimated matrices $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ that are based on exactly the same occurrences in the training data. This refinement can be applied to characteristic words analogously.

Finally, it is also possible to group words into sets of words (as is also done in Definition 16) that we call *events* and to use indicative and characteristic events in place of words. This corresponds to adding the respective columns and rows in the matrices $\hat{F}_{Y,X}$, $\hat{F}_{zY,X}$, etc. and can be formally accomplished by a special selection of the indicator and characterizer matrices Q and C . Finding good indicative and characteristic events was the strategy adopted by early OOM learning algorithms [Jaeger, 2000b]. A further generalization of this idea of considering events in place of words is

proposed by Wingate et al. [2007]. Using such events may carry an additional advantage if the estimation of $\hat{f}(Z)$ from the available data can be performed more efficiently or accurately than computing $\hat{f}(Z) = \sum_{\bar{z} \in Z} \hat{f}(\bar{z})$.

5.2.1 A General Selection Method

We will consider only the approaches above aimed at selecting non-minimal sets of indicative and characteristic words, as these will lead to matrices $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ that retain more of the information contained in the training data. In Section 5.4.1 we will give some further motivation for this. These approaches can be combined into the following general selection method:

Remark 41. *To select indicative and characteristic words for given training data, choose the at most $Z_{\max}$ admissible words occurring most often in the training data, where a word is admissible if it has a length of at least $L_{\min}$ but not longer than $L_{\max}$ and occurs at least $O_{\min}$ times in the training data.*

In the case of (controlled) processes admissibility may be further restricted – indicated by setting U_{unique} – when selecting characteristic words to words that have no prefix of length at least $L_{\min}$ occurring at the exact same positions in the training data, or when selecting indicative words to words that have no suffix of length at least $L_{\min}$ occurring at the exact same positions in the training data.

We count the occurrences of a word in the training data as the number of occurrences as a substring in the case of (controlled) stochastic processes, while in the case of stochastic languages we count the number of occurrences as a prefix when selecting indicative words and as a suffix when selecting characteristic words.

The parameters $O_{\min}$ and $Z_{\max}$ are appealing, since they yield a choice of indicative and characteristic words that is matched to the available training data and since they allow good control over the size of the resulting matrix $\hat{F}_{Y,X}$. In the following, we explain how to compute this selection efficiently and specify this in Algorithm 5. Furthermore, we present a new proposal for setting the parameters $O_{\min}$ and $Z_{\max}$ in the case of (controlled) stochastic processes.

The key trick is to represent the training data by a *radix tree*, either for the sample words themselves in the case of stochastic languages or for the suffixes of the sample sequences in the case of (controlled) stochastic processes. In the latter case the radix tree of suffixes is called a *generalized suffix tree*. Each position in the radix tree represents a word occurring as a prefix in the represented words, i.e., as a prefix of some sample word in the case of stochastic languages or as a prefix of a suffix of a training sequence,

5. LEARNING SEQUENTIAL SYSTEMS

Algorithm 5: Selection of indicative and characteristic words

In : Training data consisting of a set $S = \{\bar{s}_k\}$ of words/sequences
Parameters $L_{\min}, L_{\max}, O_{\min}, Z_{\max}, [U_{\text{unique}}]^1$
Out: Sets X, Y of indicative and characteristic words consisting of the at most $Z_{\max}$ *admissible* words occurring most often in the training data², where a word is *admissible* if it has a length between $L_{\min}$ and $L_{\max}$, occurs at least $O_{\min}$ times [and, if requested by U_{unique} , has no prefix (suffix) of admissible length that occurs at exactly the same positions in the training data when selecting characteristic (indicative) words]¹.

case *stochastic language* **do**

$\mathcal{T} \leftarrow$ radix tree for S
 $X \leftarrow \text{select_words_from_tree}(\mathcal{T})$
 $\mathcal{T} \leftarrow$ radix tree for $\text{reversed}(S)$
 $Y \leftarrow \text{reversed}(\text{select_words_from_tree}(\mathcal{T}))$

case (*controlled*) *stochastic process* **do**

$\mathcal{T} \leftarrow$ generalized suffix tree for S
 $Y \leftarrow \text{select_words_from_tree}(\mathcal{T})$
if U_{unique} **then**
 $\mathcal{T} \leftarrow$ generalized suffix tree for $\text{reversed}(S)$
 $X \leftarrow \text{reversed}(\text{select_words_from_tree}(\mathcal{T}))$
else $X \leftarrow Y$

Procedure $\text{reversed}(Z)$

return $Z^r = \{\text{reversed word/sequence } \bar{z}^r \mid \bar{z} \in Z\}$

Procedure $\text{select_words_from_tree}(\mathcal{T})$

Annotate all nodes of $\mathcal{T}$ by their leaf counts in one depth-first pass
 $\mathcal{N} \leftarrow \{\text{root of } \mathcal{T}\}$ // priority queue of candidate nodes
 $Z \leftarrow \{\}$
while $\mathcal{N} \neq \emptyset$ **do**
 $n \leftarrow$ node in $\mathcal{N}$ with largest leaf count; $\mathcal{N} \leftarrow \mathcal{N} \setminus \{n\}$ // $n \leftarrow \text{pop}(\mathcal{N})$
 if $\text{leaf_count}(n) < O_{\min}$ **then return** Z
 $E \leftarrow \{\bar{x} \text{ represented by a position on } \text{Edge}(n) \mid L_{\min} \leq |\bar{x}| \leq L_{\max}\}$
 [if U_{unique} **then** $E \leftarrow \{\text{shortest word from } E\}]^1$
 foreach $\bar{x} \in E$ **do** // E may be empty
 $Z \leftarrow Z \cup \{\bar{x}\}$
 if $|Z| = Z_{\max}$ **then return** Z
 if $|\text{word represented by } n| < L_{\max}$ **then** $\mathcal{N} \leftarrow \mathcal{N} \cup \{\text{children of } n\}$
return Z

¹[...] is only applicable in the case of (controlled) stochastic processes.

²In the case of (controlled) processes, we count the number of occurrences as a substring in the training sequences, while in the case of stochastic languages, we count the number of occurrences as a prefix (suffix) of a word in the training set when considering indicative (characteristic) words.

5.2 Choosing Indicative and Characteristic Words

also known as a substring, in the case of (controlled) processes. The occurrence counts are given by the number of leaves below the position in the radix tree, and all positions on an edge of the radix tree correspond to words occurring at the same positions in the training data. If we instead need to represent the suffixes (of training words or of prefixes of the training sequences) we can simply use a radix tree / generalized suffix tree for the reversed training data. Extracting the desired word selection from the radix tree representation of the training data via the procedure `select_words_from_tree` is straight forward. To see this, note that all descendants of a node in the radix tree represent longer words with lower leaf counts, and that the procedure therefore considers the relevant nodes in the tree $\mathcal{T}$ in decreasing order of their leaf counts, where we call a node relevant if any position on its edge represents a word of length less than or equal to $L_{\max}$. We should note that satisfying the U_{unique} requirement for indicative sequences does not really require a suffix tree of the reversed training data (as used in Algorithm 5 for simplicity) but can also be computed from the suffix tree for the training data by making use of suffix links.

If the training data consists of a total of N symbols, the radix tree / generalized suffix tree consists of at most $2N$ nodes and can be computed and annotated in $\mathcal{O}(N)$ time. The priority queue $\mathcal{N}$ can be implemented such that retrieving and removing the largest node as well as inserting a new node each take at most $\mathcal{O}(\log |\mathcal{N}|)$ time. Furthermore, the selection of words from the $Edge(n)$ takes constant time for every word selected, or constant time if no word is selected. Since the procedure `select_words_from_tree` considers every node in $\mathcal{N}$ at most once, and the size of $\mathcal{N}$ is bounded by the number of nodes in the radix tree, this gives a runtime complexity for the procedure `select_words_from_tree` of $\mathcal{O}(N \log N + |Z|)$, where Z is the returned set of words. The complexity of Algorithm 5 is therefore in $\mathcal{O}(N \log N + |X| + |Y|)$, where X and Y are the returned sets of indicative and characteristic words. If $L_{\min} = 0$ we can give a different bound, since in this case nodes (at most $|\Sigma|$ at once) can only be added to $\mathcal{N}$ if at least one word is added to Z . Therefore, at most $|\Sigma||Z|$ nodes are ever added to $\mathcal{N}$ during the run of the procedure `select_words_from_tree`, where Z is the returned set of words, giving a different runtime complexity bound of $\mathcal{O}(|\Sigma||Z| \log |\Sigma||Z|)$ for the procedure `select_words_from_tree`.

Next, we propose a new approach to selecting indicative and characteristic words in the case of (controlled) stochastic processes. We will allow words of arbitrary length, i.e., we set $L_{\min} = 0$ and $L_{\max} = \infty$, and use the parameter $Z_{\max}$ only to bound the size of the resulting matrix $\hat{F}_{Y,X}$ such that it remains numerically tractable. For instance, for the empirical investigations in Chapter 6, we will generally use a setting

5. LEARNING SEQUENTIAL SYSTEMS

of $Z_{\max} = 1024$. The parameter that we rely on to obtain words that represent the available training data well is the parameter $O_{\min}$, for which we must find a suitable setting that takes the size N of the available training data as well as the alphabet sizes Σ_O and Σ_I into account. We propose the following heuristic for setting the parameter $O_{\min}$.

The basic idea is to require that each entry in the matrix $\hat{F}_{Y,X}$ should be based on a certain number of unique occurrences in the training data. Since words $\bar{x}\bar{y}$ corresponding to entries $\hat{f}(\bar{x}\bar{y})$ have different occurrence probabilities, we require that such a word $\bar{x}\bar{y}$ should occur approximately $f(\bar{y}|\bar{x})$ times in the training data. As we do not know anything about the process, we simply assume a uniformly random process, i.e., that $f(\bar{y}|\bar{x}) = |\Sigma_O|^{-|\bar{y}|}$ for all $\bar{x}\bar{y} \in \Sigma^*$. If we now assume that X and Y consist of all words of a fixed length l , i.e., that $\hat{F}_{Y,X}$ is of size $|\Sigma|^l \times |\Sigma|^l$, then these assumptions let us compute the length l , which we will call the *effective word length*, by summing the occurrence counts for all entries, which gives $|\Sigma|^{2l}|\Sigma_O|^{-l}$, and equating this with the number N of unique positions in the training data, since different words of the same fixed length must occur at different positions. This gives

$$N = |\Sigma|^{2l}|\Sigma_O|^{-l} = |\Sigma|^l \cdot |\Sigma_I|^l \quad \Rightarrow \quad l = \frac{\log N}{\log |\Sigma| + \log |\Sigma_I|}.$$

Next, to obtain the required occurrence counts $O_{\min}$ for an indicative (or characteristic) word $\bar{x}$, we would like to just sum the occurrence counts for the entries in the column $[\hat{F}_{Y,X}]_{\bar{x}}$ indexed by $\bar{x}$. However, we note that the selection of indicative and characteristic words based on the minimum occurrence count $O_{\min}$ has the property that if a word $\bar{x}$ is selected, i.e., $\#(\bar{x}) \geq O_{\min}$, then all of its subsequences will also be selected, as these have the same or a higher occurrence count. In fact, if we stick to the assumption that any word $\bar{x}\bar{y}$ with $|\bar{y}| = l$ occurs $|\Sigma_O|^{-l}$ times, then a word $\bar{x}\bar{z}$ with $|\bar{z}| = k < l$ will occur $|\Sigma|^{l-k} \cdot |\Sigma_O|^{-l}$ times, since it will occur as a prefix of $|\Sigma|^{l-k}$ words of length $|\bar{x}| + l$, each of which occurs at $|\Sigma_O|^{-l}$ distinct positions.

We will therefore consider the matrix $\hat{F}_{Y,X} = \hat{F}_{\Sigma^{\leq l}, \Sigma^{\leq l}}$ for sets X and Y consisting of all words of length equal to or less than the effective word length l instead. We wish to uphold the basic requirement that every entry in the matrix $\hat{F}_{Y,X}$ should be based on its own share of unique occurrences in the training data, and therefore obtain $O_{\min}$ by summing the assumed occurrence counts for the entries in any column of the matrix $\hat{F}_{\Sigma^{\leq l}, \Sigma^{\leq l}}$, which gives

$$O_{\min} = \sum_{\bar{y} \in \Sigma^{\leq l}} |\Sigma|^{l-|\bar{y}|} |\Sigma_O|^{-l} = \sum_{k=0}^l |\Sigma|^k |\Sigma|^{l-k} |\Sigma_O|^{-l} = (l+1) |\Sigma_I|^l.$$

Clearly, some of these assumptions are rather ad hoc, especially requiring that a word $\bar{x}\bar{y}$ of length $2l$ should occur approximately $f(\bar{y}|\bar{x})$ times in the training data. However, this does lead to a formula for $O_{\min}$ that seems to work well for all the benchmark examples we consider in Chapter 6.

5.3 Determining the Model Dimension

The model dimension determines the number of parameters of the model and thereby the model complexity. We can approach the task of selecting a suitable model dimension in two different ways. First of all, we may be interested in estimating the rank of the external function f and use this as the model dimension. This may be the case if it is known that the external function f has a small rank that carries some meaning. On the other hand, we may rather be interested in choosing a model dimension that allows for a good approximation of the external function f from the available data, for instance when faced with real-world systems of large or possibly infinite rank. These goals are related, as one can only hope to estimate an exact model if the model dimension is at least $\text{rank}(f)$. However, with only limited training data, it may be advisable to use a lower model dimension of at most $\text{rank}(f)$ to avoid overfitting.

The default approach to selecting a suitable model dimension is to use cross-validation, but this is computationally intensive. Here, it is actually possible to estimate the rank of the external function f , which is $\text{rank}(F)$, from the estimate $\hat{F}_{Y,X}$. Since $\hat{F}_{Y,X}$ will typically have full rank due to estimation errors, we must resort to computing the *numerical rank* of $\hat{F}_{Y,X}$ instead, which measures the number of dimensions that can be distinguished from noise, as we explain in the following Section 5.3.1. The numerical rank of $\hat{F}_{Y,X}$ should give a lower bound for the rank of F that tends to $\text{rank}(F)$ as the amount of training data increases and the variance decreases. It is therefore reasonable to postulate that the numerical rank of $\hat{F}_{Y,X}$ might also be a well-suited choice for the model dimension [Jaeger, 1998].

Finally, instead of selecting a model dimension that gives the best possible model estimate for the available training data, one may instead consider a fixed model dimension d lower than $\text{rank}(F_{Y,X})$, and then strive to find a d -dimensional model approximation from the available training data. This approach may be desirable if it is important to learn small models. In this case, however, it is not at all clear if the learning procedure

5. LEARNING SEQUENTIAL SYSTEMS

from Algorithm 4 will converge to a “best” d -dimensional approximation in the limit of infinite training data. In fact, our results in Section 6.3 when estimating stochastic processes suggest that such an approach will not give a best approximation in terms of the metric we use to evaluate the learned model (cross-entropy rate). We will not deal with this separate task of “low-rank approximation”, but point out that this has recently been investigated [Kulesza et al., 2014, 2015a].

5.3.1 Numerical Rank

For suitably chosen indicative and characteristic words, one can expect to have $\text{rank}(f) = \text{rank}(F_{Y,X})$. However, since one only has access to an estimate $\hat{F}_{Y,X}$ of this matrix, a typical approach is to determine what is known as the *numerical rank* (or *effective rank* or *pseudorank*). We give a brief description following Hansen [1998].

The *numerical ε -rank* r_ε of a matrix A may be defined as the smallest rank of any matrix that can be obtained from A by a small perturbation E of size at most ε :

$$r_\varepsilon(A) = \min_{\|E\| \leq \varepsilon} \text{rank}(A + E).$$

In terms of the singular values $\sigma_1 \geq \dots \geq \sigma_K$ of A this means that r_ε satisfies $\sigma_{r_\varepsilon} > \varepsilon \geq \sigma_{r_\varepsilon+1}$ if the size of the perturbation E is measured by the spectral norm $\|\cdot\|_2$ or, alternatively, that r_ε is the smallest k such that $\sum_{i=k+1}^K \sigma_i^2 \leq \varepsilon$ if the squared Frobenius norm $\|\cdot\|_F^2$ is used instead. Both criteria can be used to determine r_ε .

The *numerical rank* of A is then $r_\varepsilon(A)$ for some reasonable choice of ε . In the case that A is a noisy estimate of an underlying matrix $\tilde{A}$, one should choose ε to be of the same size as the expected error, i.e., $\varepsilon \approx \mathbb{E}[\|A - \tilde{A}\|]$.

Note that this notion of numerical rank assumes that the errors on matrix entries of A are of comparable magnitudes and can be reasonably quantified, and that there is a significant *gap* between σ_{r_ε} and $\sigma_{r_\varepsilon+1}$. Otherwise, the numerical rank is somewhat arbitrary. It is furthermore important to note that the numerical rank essentially measures how many dimensions can be distinguished from noise and therefore may underestimate the true rank of the underlying matrix.

The main difficulty in determining the numerical rank of the matrix $\hat{F}_{Y,X}$ lies in finding a suitable ε . This may be approached by obtaining estimates for, or bounds on, the variances of the individual matrix entries [Jaeger, 1998; James and Singh, 2004], which may, however, differ widely across $\hat{F}_{Y,X}$.

Here, we will use the variance estimates $\hat{\mathbb{V}}_{0,0}[\hat{f}(\bar{x})]$ that we have established in Section 5.1.1. Using parameters $K = 0$ and $M = 0$ means that we do not introduce

5.3 Determining the Model Dimension

any bias towards overestimating the variances. Also, these estimates have the property that $\hat{\mathbb{V}}_{0,0}[\hat{f}(\bar{x})] = 0$ whenever $\hat{f}(\bar{x}) = 0$, which is typically the case for many of the entries in $\hat{F}_{Y,X}$, and we have found this to be crucial to avoid overly conservative, i.e., too low, rank estimates. The matrix $\hat{V}_{Y,X}$ of element-wise variance estimates for the entries in $\hat{F}_{Y,X}$ that we use is therefore

$$\hat{V}_{Y,X} = \left[\hat{\mathbb{V}}_{0,0}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{y} \in Y, \bar{x} \in X}.$$

Furthermore, we will apply a scaling to the rows and columns of $\hat{F}_{Y,X}$ (and scale the corresponding variance estimates accordingly) in an attempt to normalize the variances to be of similar magnitude. For this, define

$$v_Y^\top = \frac{1}{|X|} \left[\sum_{\bar{x} \in X} \hat{\mathbb{V}}_{2,0}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{y} \in Y}^\top, \quad \text{and} \quad v_X = \frac{1}{|Y|} \left[\sum_{\bar{y} \in Y} \hat{\mathbb{V}}_{2,0}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{x} \in X},$$

which are just the row and column averages of the element-wise variances, with the exception that here we employ a slight bias to avoid zero variance estimates, since we will use the element-wise inverse square root of these vectors to scale $\hat{F}_{Y,X}$. Let

$$\tilde{F}_{Y,X} = \mathbf{D}(v_Y^\top)^{-\frac{1}{2}} \hat{F}_{Y,X} \mathbf{D}(v_X)^{-\frac{1}{2}}, \quad \text{and} \quad \tilde{V}_{Y,X} = \mathbf{D}(v_Y^\top)^{-1} \hat{V}_{Y,X} \mathbf{D}(v_X)^{-1}.$$

Now $\tilde{F}_{Y,X}$ is a row and column scaled version of $\hat{F}_{Y,X}$ with corresponding variance estimates given by $\tilde{V}_{Y,X}$, where we have essentially divided each row and column of $\hat{V}_{Y,X}$ by its average. We will compute the spectrum $\sigma_1 \geq \dots \geq \sigma_K$ and numerical rank of $\tilde{F}_{Y,X}$ instead of $\hat{F}_{Y,X}$. The justification is that the underlying rank of the error-free matrix $F_{Y,X}$ will not change under this row and column scaling. To skip this novel normalization step, we can simply set $v_Y^\top = v_X = 1$.

Next, based on a recipe by Golub and Van Loan [1996, Section 5.5.8], Jaeger [1998] suggests to use the average relative error of the matrix entries multiplied by the largest matrix entry as cutoff ε_J , which is therefore computed as

$$\varepsilon_J = \|\tilde{F}_{Y,X}\|_\infty \frac{\sum_{\bar{y} \in Y, \bar{x} \in X} \alpha_{\bar{y},\bar{x}}}{|Y| \cdot |X|}, \quad \text{where } \alpha_{\bar{y},\bar{x}} = \begin{cases} \frac{\sqrt{[\tilde{V}_{Y,X}]_{\bar{y},\bar{x}}}}{[\tilde{F}_{Y,X}]_{\bar{y},\bar{x}}} & \text{for } [\tilde{F}_{Y,X}]_{\bar{y},\bar{x}} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

and to compute the numerical rank r_{ε_J} as the largest k such that $\sigma_k > \varepsilon_J$.

Instead, we propose several alternative cutoffs based on the above derivation of the numerical rank that suggests using $\varepsilon \approx \mathbb{E}[\|\tilde{E}_{Y,X}\|]$, where we assume the errors $\tilde{E}_{Y,X}$

5. LEARNING SEQUENTIAL SYSTEMS

to have zero mean and element-wise variance given by $\tilde{V}_{Y,X}$.

In the case that we use the squared Frobenius norm to measure the error, we may simply use $\varepsilon_F = \mathbb{E}[\|\tilde{E}_{Y,X}\|_F^2] = \|\tilde{V}_{Y,X}\|_F^2$ due to the linearity of the expectation and then compute the numerical rank r_{ε_F} as the smallest k such that $\sum_{i=k+1}^K \sigma_i^2 \leq \varepsilon_F$.

However, if we use the spectral norm, it is not clear how we should compute $\mathbb{E}[\|\tilde{E}_{Y,X}\|_2]$, because this depends on the actual distribution of the errors and their correlations. One approach is to assume the errors to be normally distributed and uncorrelated. We can then approximate the expectation by sampling, where few samples, for instance five, seem to suffice. This yields a cutoff ε_S from which we compute the numerical rank r_{ε_S} as the largest k such that $\sigma_k > \varepsilon_S$.

Alternatively, we can make the crude assumption that the errors in $\tilde{E}_{Y,X}$ all have the same sign and size that is given by their average standard deviation. Using the spectral norm to measure the error, this yields as cutoff

$$\varepsilon_\emptyset = \frac{1}{\sqrt{|Y| \cdot |X|}} \sum_{\tilde{y} \in Y, \tilde{x} \in X} \sqrt{[\tilde{V}_{Y,X}]_{\tilde{y}, \tilde{x}}}$$

and as numerical rank $r_{\varepsilon_\emptyset}$ the largest k such that $\sigma_k > \varepsilon_\emptyset$.

We have observed that in many cases the cutoff $\varepsilon_\emptyset$ is too small, while the cutoff ε_S is slightly too large, and therefore propose to use the geometric mean

$$\varepsilon_2 = \sqrt{\varepsilon_S \varepsilon_\emptyset}$$

as cutoff when measuring the error by the spectral norm and to compute the numerical rank r_{ε_2} as the largest k such that $\sigma_k > \varepsilon_2$. This appears to be a good compromise that gives excellent results on the benchmark datasets we consider later in Section 6.3.

Independent of such error estimates, it may be reasonable to assume that there will be a relative “gap” between σ_{r+1} and σ_r in the singular value spectrum of $\tilde{F}_{Y,X}$ around the true rank $r = \text{rank}(F_{Y,X})$. Bailly et al. [2009] propose to search for such a gap by maximizing the second order slope of the singular values logarithm curve starting from σ_{r_ε} , where r_ε is a numerical rank similar to r_{ε_F} , i.e., they estimate the rank r_{gap} as

$$r_{\text{gap}} = \arg \max_{r \geq r_{\varepsilon_F}} \left\{ \frac{\sigma_r \sigma_{r+2}}{\sigma_{r+1}^2} \right\}.$$

See Section 6.3 for an empirical investigation of these approaches to estimating a suitable model dimension based on the numerical rank of $\hat{F}_{Y,X}$.

Finally, let us examine the computational complexity of finding the numerical rank

5.4 Selecting the Characterizer and Indicator

of the matrix $\hat{F}_{Y,X}$ or, alternatively, its scaled version $\tilde{F}_{Y,X}$. Naively, we could obtain its full spectrum $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_K$ from its singular value decomposition (SVD) $USV^\top$, which requires time in $\mathcal{O}(|X||Y| \min\{|X|, |Y|\})$ to compute. However, for all numerical ranks r_ε other than r_{gap} , it suffices to compute the d -truncated SVD $U_d S_d V_d^\top$ for some $d > r_\varepsilon$, since $\sum_{k=d+1}^K \sigma_k^2 = \|USV - U_d S_d V_d^\top\|_F^2$, which can be computed from the d -truncated SVD in time $\mathcal{O}(d|X||Y|)$. As we do not know r_ε and therefore the required d , we may compute d -truncated SVDs for increasing $d \in \{2^k \mid k \in \mathbb{N}\}$, until the cutoff criterium for the given ε is satisfied. As the runtime cost of computing the d -truncated SVD is only $\mathcal{O}(d|X||Y|)$, this approach has a runtime complexity of only $\mathcal{O}(|X||Y| \sum_{k=1}^{\lceil \log_2 r_\varepsilon \rceil} 2^k) = \mathcal{O}(|X||Y|r_\varepsilon)$, where r_ε is the computed numerical rank. In fact, there are specialized and efficient algorithms for the basic task of finding a low-rank approximation to a matrix for a given error tolerance ε [Halko et al., 2011].

5.4 Selecting the Characterizer and Indicator

The effect of the characterizer C and indicator Q is to reduce the available data in $\hat{F}_{Y,X}$, $\hat{F}_{zY,X}$, $\hat{F}_Y^\top$, and $\hat{F}_X$ to a d -dimensional representation, where d is the chosen target dimension for the model to be learned. Assuming that $d = \text{rank}(F) = \text{rank}(CF_{Y,X}Q)$, then the matrices $CF_{Y,X}Q$, $CF_{zY,X}Q$, $CF_Y^\top$, and $F_X Q$ together contain the same information as F and are sufficient to reconstruct a SS model for f via the learning Equations (2). The requirement that $CF_{Y,X}Q$ must have full rank d , therefore, ensures that no information is lost.

In fact – provided that $CF_{Y,X}Q$ has full rank d – really any choice of characterizer and indicator *may* be used and will lead to a consistent model estimation, i.e., a correct model will be obtained in the limit of infinite training data. Hamilton et al. [2013] show that for certain dynamical systems a random choice of characterizer C does indeed work well. However, in general the choice of characterizer C and indicator Q is central to achieving statistical efficiency, i.e., making efficient use of the available training data. This step lies at the heart of the learning procedure, and in fact much research – even if not explicitly stated – can be seen as optimizing this step of the learning algorithm.

Before we explore various methods for selecting the characterizer and indicator matrices C and Q , we will briefly note some properties of the selection of C and Q in the following Lemma 42, which follows trivially from the learning Equations (2). Lemma 42 also highlights that the roles of C and Q are not fully symmetrical.

Lemma 42. *Let $C \in \mathbb{R}^{d \times |Y|}$ and $Q \in \mathbb{R}^{|X| \times d}$ be characterizer and indicator matrices with training data represented in $\hat{F}_{Y,X}$, $\hat{F}_{zY,X}$, $\hat{F}_Y^\top$, and $\hat{F}_X$, and let $\rho \in \mathbb{R}^{d \times d}$ be a non-*

5. LEARNING SEQUENTIAL SYSTEMS

singular matrix. Applying the learning Equations (2) for (C, Q) and (C', Q) , where $C' = \rho C$, will lead to equivalent models $\hat{\mathcal{M}}$ and $\hat{\mathcal{M}}'$, respectively, that are related by $\hat{\mathcal{M}}' = \rho \hat{\mathcal{M}} \rho^{-1}$, while applying the learning Equations (2) for (C, Q) and (C, Q') , where $Q' = Q \rho$, will lead to identical models $\hat{\mathcal{M}} = \hat{\mathcal{M}}'$.

5.4.1 Selection or Grouping of Rows and Columns of $\hat{F}$

It is important to note that the choice of indicative and characteristic words discussed in Section 5.2 can be viewed equivalently as a special choice of characterizer and indicator. To see this, assume one could estimate the entire matrix $\hat{F}$ from data. Then any selection of rows indexed by Y and columns indexed by X from $\hat{F}$ can be achieved by characterizer and indicator matrices C, Q of the form $C = C' C^Y$ and $Q = Q^X Q'$, where C^Y and Q^X are appropriate binary matrices with a single one entry per row and column, and zeros otherwise, such that $C^Y \hat{F} Q^X = \hat{F}_{Y,X}$. This can easily be extended to account for groupings of words into events by allowing several one entries per row for C^Y or per column for Q^X .

One advantage of this point of view is that this immediately justifies the optional grouping of words into events mentioned in Section 5.2. But more importantly, this highlights that choosing indicative and characteristic words as described in Section 5.2 is, in fact, a restricted approach to the more general problem of finding appropriate characterizer and indicator matrices. We argue that a good choice of characterizer and indicator is the key to achieving high statistical efficiency of the learning procedure and that therefore the (pre-)selection of indicative and characteristic words should be guided by trying to retain as much information from the available training data as possible. In other words, the (pre-)selection of indicative and characteristic words in Section 5.2 is primarily a practical necessity that should rather be seen as discarding rows and columns from $\hat{F}$ that carry only little or no information.

5.4.2 Spectral Methods

Spectral learning algorithms for SSs can often be seen as a specific approach to selecting the characterizer and indicator matrices C and Q and therefore perfectly fit the learning framework presented here. Nevertheless, to understand the motivation for spectral methods, we will first consider them as a learning method for SSs in their own right and therefore briefly return to first principles from Chapter 3.

Let us assume we want to construct a SS model for a given function $f : \Sigma^* \rightarrow \mathbb{R}$. Recall that the columns indexed by the word $\bar{x}$ of the matrices F and F_z correspond

5.4 Selecting the Characterizer and Indicator

to the functions $f_{\bar{x}}$ and $f_{\bar{x}z}$, and that the operator τ_z of any minimal model $\mathcal{M}$ for f – regarded as a linear operator $\tilde{\tau}_z$ on the space $\mathcal{F}$ – satisfies $\tilde{\tau}_z(f_{\bar{x}}) = f_{\bar{x}z}$ (cf. Proposition 3). The matrix τ_z is just a representation of this operator with respect to some basis of $\mathcal{F}$. We can therefore regard the columns of $F = [f_{\bar{x}}]_{\bar{x} \in \Sigma^*}$ and $F_z = [f_{\bar{x}z}]_{\bar{x} \in \Sigma^*}$ as argument-value pairs for the operator $\tilde{\tau}_z$, from which we can recover $\tilde{\tau}_z$. To obtain a matrix representation τ_z , we first need to fix some basis for the column space $\mathcal{F}$ and map the columns of F and F_z to their coordinate representation in $\mathbb{R}^d$ with respect to this basis. We can then use these coordinate representations as argument-value pairs for the matrix representation τ_z . The remaining model parameters σ and ω_ε are obtained similarly. This approach means we must perform the following steps:

- (i) Find a basis B for the column space of F , i.e., *identify* the space $\mathcal{F}$.
- (ii) Map the columns $f_{\bar{x}}$ and $f_{\bar{x}z}$ of F and F_z to coordinate vectors $a_{\bar{x}}$ and $a_{\bar{x}z}$ in $\mathbb{R}^d$ with respect to this basis B . Collect these into the matrices $A = [a_{\bar{x}}]_{\bar{x} \in \Sigma^*}$ and $A_z = [a_{\bar{x}z}]_{\bar{x} \in \Sigma^*}$, respectively. Also, compute ω_ε as the coordinate representation of the column vector f_ε of F with respect to the basis B .
- (iii) Compute τ_z from the argument and value matrices A and A_z , i.e., solve $\tau_z A = A_z$, and compute σ by solving the equation $\sigma A = f_\varepsilon^\top$.

In practice, however, we face two problems. First of all, we can only deal with the finite matrices $F_{Y,X}, F_{zY,X}, F_Y^\top$, and F_X . However, provided that $\text{rank}(F_{Y,X}) = \text{rank}(F) = d$, the column spaces are isomorphic, which means that this is not an issue. Secondly, we can only obtain estimates $\hat{F}_{Y,X}, \hat{F}_{zY,X}, \hat{F}_Y^\top$, and $\hat{F}_X$. The spectral methods therefore take the following approach:

- (i) Estimate the column space of $F_{Y,X}$ by the column space of the best (rank- d) approximation $\tilde{F}_{Y,X}$ to the matrix $\hat{F}_{Y,X}$ in the Frobenius norm and fix a basis B for the column space of $\tilde{F}_{Y,X}$.
- (ii) Project the columns of $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ orthogonally onto the estimated column space and find their coordinate representations $\hat{A}$ and $\hat{A}_z$ w.r.t. the basis B . Set $\hat{\omega}_\varepsilon$ to the coordinate representation w.r.t. B of the orthogonal projection of $\hat{F}_Y^\top$ onto the estimated column space.
- (iii) Compute the matrices $\hat{\tau}_z$ by solving $\hat{\tau}_z \hat{A} \approx \hat{A}_z$ using ordinary least squares, and compute $\hat{\sigma}$ by solving $\hat{\sigma} \hat{A} \approx \hat{F}_X$ using ordinary least squares.

5. LEARNING SEQUENTIAL SYSTEMS

Step (i) is accomplished by computing the SVD of $\hat{F}_{Y,X} = USV^\top$ and considering U_d that is formed by the first d columns of U , S_d that consists of only the first d rows and columns of S , and V_d that consists of the first d columns of V . The d -truncated SVD $U_d S_d V_d^\top$ is then the best rank- d approximation to $\hat{F}_{Y,X}$ by the Eckart-Young theorem [Eckart and Young, 1936]. The columns of U_d form an orthonormal basis of the d -dimensional column space of $U_d S_d V_d^\top$ called the *principal subspace* of $\hat{F}_{Y,X}$, which is used as an estimate of the column space of $F_{Y,X}$.

For Step (ii) we first note that the orthogonal projection onto the principal subspace spanned by U_d is given by $U_d(U_d^\top U_d)^{-1}U_d^\top = U_d U_d^\top$. The coordinate representation (in terms of the basis U_d) of the projected columns of $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ as well as $\hat{F}_Y^\top$ is therefore given by $\hat{A} = U_d^\top \hat{F}_{Y,X}$, $\hat{A}_z = U_d^\top \hat{F}_{zY,X}$, and $\hat{\omega}_\varepsilon = U_d^\top \hat{F}_Y^\top$, respectively. Note that $U_d^\top \hat{F}_{Y,X} = U_d^\top U S V^\top = S_d V_d^\top$, as $U_d^\top U = [\mathbf{I}_d, 0]$ due to the orthonormality of U .

For Step (iii) the matrices $\hat{\tau}_z$ are computed by solving $\hat{\tau}_z U_d^\top \hat{F}_{Y,X} \approx U_d^\top \hat{F}_{zY,X}$ in the least squares sense, i.e., by computing $\hat{\tau}_z = U_d^\top \hat{F}_{zY,X} (U_d^\top \hat{F}_{Y,X})^\dagger = U_d^\top \hat{F}_{zY,X} V_d S_d^\dagger$. Similarly, $\hat{\sigma}$ is computed by solving $\hat{\sigma} U_d^\top \hat{F}_{Y,X} \approx \hat{F}_X$ in the least squares sense, i.e., by computing $\hat{\sigma} = \hat{F}_X (U_d^\top \hat{F}_{Y,X})^\dagger = \hat{F}_X V_d S_d^\dagger$.

The spectral learning approach is equivalent to a certain selection of the characterizer C and indicator Q , namely $C = U_d^\top$ and $Q = (C \hat{F}_{Y,X})^\dagger = V_d S_d^\dagger$, which can be seen by substituting these into the learning Equations (2) and noting $C \hat{F}_{Y,X} Q = U_d^\top U S V^\top V_d S_d^\dagger = \mathbf{I}_d$. In summary, we therefore obtain the following spectral learning Algorithm 6 for computing an appropriate characterizer C and indicator Q for use in the learning Equations (2), which has a computational complexity of $\mathcal{O}(d|X||Y|)$ that arises from the computation of the d -truncated SVD.

Algorithm 6: Spectral learning algorithm

In : Estimates $\hat{F}_{Y,X}$; the model dimension d
Out: C , Q for use in the learning Equations (2)
 $U_d, S_d, V_d^\top \leftarrow d$ -truncated SVD of $\hat{F}_{Y,X}$
 $C \leftarrow U_d^\top$, $Q \leftarrow (C \hat{F}_{Y,X})^\dagger = V_d S_d^\dagger$

We conclude with some remarks. First of all, this spectral algorithm is motivated by computing the best rank- d approximation to the matrix $\hat{F}_{Y,X}$. However, the matrix $F_{Y,X}^{\hat{\mathcal{M}}}$ reconstructed via the so learned model $\hat{\mathcal{M}}$ – which will clearly have rank at most d – will in general *not* be a best rank- d approximation to $\hat{F}_{Y,X}$. This is due to the fact that constructing $F_{Y,X}^{\hat{\mathcal{M}}}$ from the model $\hat{\mathcal{M}}$ enforces additional structure. Interestingly, we have observed that the reconstructed matrix $F_{Y,X}^{\hat{\mathcal{M}}}$ is often a better approximation

to the *true* matrix $F_{Y,X}$ than either of $\hat{F}_{Y,X}$ and its best rank- d approximation.

Furthermore, this spectral approach is sometimes referred to as principal component analysis (PCA). However, PCA typically involves mean-centering the data first. PCA therefore projects the data onto a d -dimensional *affine* subspace that contains the data mean, while here we know that the data $\hat{F}_{Y,X}$ lie approximately on a true subspace (even though they do not have zero mean). Mean-centering the data is therefore inappropriate in this context – nevertheless, it is sometimes done anyway [Bailly et al., 2009]. To avoid confusion, we refer to learning algorithms based on this idea simply as spectral learning algorithms [Rosencrantz et al., 2004; Hsu et al., 2009; Bailly et al., 2009; Siddiqi et al., 2010; Boots and Gordon, 2010; Bailly, 2011; Balle et al., 2011, 2014]. Furthermore, an online version of this spectral learning algorithm has been developed by Boots and Gordon [2011], whereas a modification that combines the subspace estimation step (determining the characterizer C) and linear regression step (solving the learning Equations (2)) into a single optimization problem is given by Balle et al. [2012].

Sample complexity results that fall into the probably approximately correct (PAC) learning framework [Valiant, 1984] are available for several spectral learning algorithms [Hsu et al., 2009; Bailly et al., 2009; Siddiqi et al., 2010; Bailly, 2011]. These give bounds on the number or size N of samples that are required to obtain a model estimate $\hat{\mathcal{M}}$ that is approximately correct (i.e., such that $|f_{\hat{\mathcal{M}}} - f| < \varepsilon$ for a given ε and a specified distance measure) with probability at least $1 - \delta$ for a given δ . Typically, the required size N is shown to be polynomial in the PAC parameters $1/\varepsilon$ and $1/\delta$, as well as other parameters that depend on f such as the alphabet size $|\Sigma|$ and the rank of f . However, the available results essentially use the L_1 distance on functions, often restricted to words of length at most T , i.e., $|f_{\hat{\mathcal{M}}} - f| = \sum_{\bar{x} \in \Sigma^{\leq T}} |f_{\hat{\mathcal{M}}}(\bar{x}) - f(\bar{x})|$. In the case of stochastic languages the functions all lie in the Hilbert space $l_2(\Sigma^*)$ and the metric of this function space may be better suited. For stochastic processes, a more natural choice may be the cross-entropy, which is related to finding a maximum likelihood estimate of the model parameters from data.

5.4.3 A Weighted Spectral Learning Algorithm

We now address a shortcoming of the spectral methods as they are presented above and are currently used. They implicitly assume that the estimates $\hat{f}(\bar{x})$ are all of similar quality, i.e., that the estimates $\hat{f}(\bar{x})$ all have the same variance. This, however, is not generally the case, which suggests that a weighted spectral learning approach with

5. LEARNING SEQUENTIAL SYSTEMS

weights reflecting the quality of the estimates may give better results.

We will assign a weight $w(\bar{x}) > 0$ to each estimate $\hat{f}(\bar{x})$ that measures the *precision* of the estimate, such that a low weight $w(\bar{x})$ signifies a low confidence in the estimate $\hat{f}(\bar{x})$, i.e., we expect it to have a large error, while a large weight means that the estimate is highly accurate. A natural and common choice for such a weight $w(\bar{x})$ is the inverse variance $\mathbb{V}[\hat{f}(\bar{x})]^{-1}$ of the estimate $\hat{f}(\bar{x})$. For this, we will estimate the variance as presented in Section 5.1.1 and then use the following weights:

$$w(\bar{x}) = \hat{\mathbb{V}}_{2,3}[\hat{f}(\bar{x})]^{-1}.$$

Furthermore, we disregard the covariance structure among the estimates $\hat{f}(\bar{x})$, as this is difficult to estimate reliably from the data. That is, we make the simplifying assumption that the estimation errors for the estimates $\hat{f}(\bar{x})$ are uncorrelated. We collect the weights for the estimates in $\hat{F}_{Y,X}$, $\hat{F}_{zY,X}$, $\hat{F}_Y^\top$, and $\hat{F}_X$ into the corresponding weight matrices $W_{Y,X}$, $W_{zY,X}$, $W_Y^\top$, and W_X .

In the following we will show how one can modify the above spectral method to take such weights into account. For this, we first introduce some required definitions and weighted regression methods, namely weighted least squares (WLS), generalized least squares (GLS), and best weighted low-rank approximations. We then explain how these methods can be applied to make use of the weights in the spectral learning algorithm.

5.4.3.1 Weighted Regression and Low-Rank Approximation

We first define how we measure the *W-weighted norm* of a matrix M .

Definition 43. For a matrix $M \in \mathbb{R}^{m \times n}$ and a symmetric positive-definite weight matrix $W \in \mathbb{R}^{mn \times mn}$ with decomposition $W = LL^\top$, e.g. its Cholesky decomposition, we denote by $\|M\|_W^2$ the squared *W-weighted norm*

$$\|M\|_W^2 = \|\mathbf{vec}(M)\|_W^2 = \mathbf{vec}(M)^\top W \mathbf{vec}(M) = \|L^\top \mathbf{vec}(M)\|_F^2.$$

Using this, we can introduce a generalization to ordinary least squares linear regression that takes weights into account.

Definition 44. The generalized least squares (GLS) problem $BA \approx M \in \mathbb{R}^{m \times n}$ with symmetric positive-definite weight matrix $W \in \mathbb{R}^{mn \times mn}$ is solved for A when B and M

5.4 Selecting the Characterizer and Indicator

are given or, respectively, for B when A and M are given, by

$$\begin{aligned}\tilde{A} &= \underset{A}{\operatorname{argmin}} \|M - BA\|_W^2, \text{ or respectively} \\ \tilde{B} &= \underset{B}{\operatorname{argmin}} \|M - BA\|_W^2.\end{aligned}$$

The case where W is diagonal is called weighted least squares (WLS), while the case when $W = \mathbf{I}$ is just ordinary least squares (OLS).

This is motivated by the linear model $M = BA + E$, where M is observed with errors that are normally distributed according to $\mathbf{vec}(E) \sim \mathcal{N}(0, \lambda W^{-1})$, i.e., zero mean with covariance λW^{-1} , where λ is some positive scalar. If we are solving for A , we regard B as given and A as model parameters that we wish to estimate from a sample M . We see that $\tilde{A}$ is then the maximum likelihood estimate for A , since

$$\begin{aligned}\mathcal{L}(A | M) &= \mathbb{P}(M | A) \\ &= \mathcal{N}(\mathbf{vec}(M - BA) | 0, \lambda W^{-1}) \\ &= K \exp \left\{ -\frac{1}{2} \mathbf{vec}(M - BA)^\top \lambda^{-1} W \mathbf{vec}(M - BA) \right\} \\ &= K \exp \left\{ -\frac{1}{2} \lambda^{-1} \|M - BA\|_W^2 \right\},\end{aligned} \tag{6}$$

where K is a positive normalizing constant, and we can see that this is maximized by minimizing $\|M - BA\|_W^2$. The same is true if we solve for B , i.e., we regard B as the model parameters and A as given.

While the OLS problem assumes all matrix entries of M to be uncorrelated and to have the same variance, the WLS problem takes the element-wise variances into account but still assumes uncorrelatedness, and the GLS problem additionally takes the full covariance relations into account.

In the following we will explain how to solve a GLS problem $BA \approx M \in \mathbb{R}^{m \times n}$ with $B \in \mathbb{R}^{m \times d}$ and $A \in \mathbb{R}^{d \times n}$ by reducing it to an OLS problem or to a set of OLS problems. If the GLS problem is *overdetermined*, which we take to mean that $\operatorname{rank}(B) = d$ when solving for A or $\operatorname{rank}(A) = d$ when solving for B , a unique solution exists that can be expressed in a closed form and can be computed efficiently by solving the normal equations using a Cholesky decomposition. Note that this is a slight abuse of the term “overdetermined”, which usually merely requires $m \geq d$ when solving for A or $n \geq d$ when solving for B .

Let us begin by stating how an OLS problem can be solved via the normal equations,

5. LEARNING SEQUENTIAL SYSTEMS

which we presume to be known.

Proposition 45. *A solution $\tilde{B}$ (respectively $\tilde{A}$) to the OLS problem $BA \approx M \in \mathbb{R}^{m \times n}$ is given by solving the normal equations*

$$\begin{aligned} B(AA^\top) &= MA^\top && \text{for } B, \text{ or respectively,} \\ (B^\top B)A &= B^\top M && \text{for } A. \end{aligned}$$

In the case of an overdetermined OLS problem, the solution is unique and can be expressed in closed form as

$$\begin{aligned} \tilde{B} &= MA^\top(AA^\top)^{-1} && \text{when solving for } B, \text{ or} \\ \tilde{A} &= (B^\top B)^{-1}B^\top M && \text{when solving for } A. \end{aligned}$$

We can solve the GLS problem $BA \approx M$ with weights W by reducing it to an OLS problem as follows.

Proposition 46. *A solution $\tilde{B}$ (respectively $\tilde{A}$) to the GLS problem $BA \approx M \in \mathbb{R}^{m \times n}$ with symmetric positive-definite weight matrix $W \in \mathbb{R}^{mn \times mn}$ is given by solving the OLS problem*

$$\begin{aligned} L^\top(A^\top \otimes \mathbf{I}_m) \mathbf{vec}(B) &\approx L^\top \mathbf{vec}(M) && \text{for } B, \text{ or respectively,} \\ L^\top(\mathbf{I}_n \otimes B) \mathbf{vec}(A) &\approx L^\top \mathbf{vec}(M) && \text{for } A, \end{aligned} \tag{7}$$

where $LL^\top = W$ is the Cholesky decomposition of W . In the case of an overdetermined problem, the unique solution may be expressed in closed form as

$$\begin{aligned} \mathbf{vec}(\tilde{B}) &= ((A^\top \otimes \mathbf{I}_m)^\top W (A^\top \otimes \mathbf{I}_m))^{-1} (A^\top \otimes \mathbf{I}_m)^\top W \mathbf{vec}(M) && \text{when solving for } B, \\ \mathbf{vec}(\tilde{A}) &= ((\mathbf{I}_n \otimes B)^\top W (\mathbf{I}_n \otimes B))^{-1} (\mathbf{I}_n \otimes B)^\top W \mathbf{vec}(M) && \text{when solving for } A. \end{aligned} \tag{8}$$

Proof. This result is easily obtained by *vectorizing* the problem and using the identity $\mathbf{vec}(BA) = (\mathbf{I}_n \otimes B) \mathbf{vec}(A) = (A^\top \otimes \mathbf{I}_m) \mathbf{vec}(B)$:

$$\begin{aligned} \|M - BA\|_W^2 &= \|L^\top \mathbf{vec}(M - BA)\|_F^2 \\ &= \|L^\top \mathbf{vec}(M) - L^\top \mathbf{vec}(BA)\|_F^2 \\ &= \|L^\top \mathbf{vec}(M) - L^\top(\mathbf{I}_n \otimes B) \mathbf{vec}(A)\|_F^2 \\ &= \|L^\top \mathbf{vec}(M) - L^\top(A^\top \otimes \mathbf{I}_m) \mathbf{vec}(B)\|_F^2. \end{aligned}$$

This shows that $\tilde{A}$ and $\tilde{B}$ are given by the OLS solution to the problems from Equa-

5.4 Selecting the Characterizer and Indicator

tion (7). The expressions for the solutions $\tilde{B}$ and $\tilde{A}$ given in Equation (8) arise from the normal equations for the corresponding OLS problems. ■

Oftentimes, the weight matrix W will have a special structure that leads to special cases of the above GLS problem with simpler solutions. For instance, if $W = \mathbf{I}$ we have a usual OLS problem. We will examine the case where W is block-diagonal, corresponding to uncorrelated columns in M , and the case of WLS where W is diagonal, corresponding to uncorrelated entries in M .

Proposition 47. *A solution $\tilde{A}$ to the GLS problem $BA \approx M \in \mathbb{R}^{m \times n}$ with symmetric positive-definite block-diagonal weight matrix $W = \mathbf{D}(W_1, \dots, W_n)$ with $W_j \in \mathbb{R}^{m \times m}$ is given by solving for each column $[A]_j$ the OLS problem*

$$L_j^\top B[A]_j \approx L_j^\top [M]_j, \quad (9)$$

where $L_j L_j^\top = W_j$ is the Cholesky decomposition of W_j . In the case of an overdetermined problem, the unique solution $\tilde{A}$ can therefore be expressed in closed form as

$$[\tilde{A}]_j = (B^\top W_j B)^{-1} B^\top W_j [M]_j \quad \text{for each column } [\tilde{A}]_j \text{ of } \tilde{A}.$$

Proof. Each matrix W_j must be symmetric and positive-definite if $\mathbf{D}(W_1, \dots, W_n)$ is, so the Cholesky decomposition of W is given by the block-diagonal $L^\top = \mathbf{D}(L_1^\top, \dots, L_n^\top)$. Therefore $\|M - BA\|_W^2 = \|L^\top \mathbf{vec}(M - BA)\|_F^2 = \sum_{j=1}^n \|L_j^\top ([M]_j - B[A]_j)\|_F^2$, so we can solve the GLS problem for each column of A independently. ■

Note that the block-diagonal structure of W does not lead to an equivalent simplification when solving for B by simply transposing the problem. This is because $\|M - BA\|_W^2 = \|\mathbf{vec}(M - BA)\|_W^2 = \|P \mathbf{vec}(M^\top - A^\top B^\top)\|_W^2 = \|M^\top - A^\top B^\top\|_{P^\top W P}^2$ for a suitable permutation matrix P , but $P^\top W P$ will no longer be block-diagonal. Nevertheless, it is still possible to exploit the block-diagonal structure of W to achieve some computational savings as follows [cf. Markovsky and Van Huffel, 2007a, Section 5].

Proposition 48. *A solution $\tilde{B}$ to the (overdetermined) GLS problem $BA \approx M \in \mathbb{R}^{m \times n}$ with symmetric positive-definite block-diagonal weight matrix $W = \mathbf{D}(W_1, W_2, \dots, W_n)$ with $W_j \in \mathbb{R}^{m \times m}$ is given by*

$$\mathbf{vec}(\tilde{B}) = \left[\sum_{j=1}^n [A]_j [A]_j^\top \otimes W_j \right]^{-1} \mathbf{vec} \left(\begin{bmatrix} W_1 [M]_1, \dots, W_n [M]_n \end{bmatrix} A^\top \right).$$

5. LEARNING SEQUENTIAL SYSTEMS

Proof. This is a simplified version of Equation (8) when solving for B . To see this, note that

$$\begin{aligned}
& (A^\top \otimes \mathbf{I}_m)^\top W (A^\top \otimes \mathbf{I}_m) \\
&= \left[[A]_1 \otimes \mathbf{I}_m, \dots, [A]_n \otimes \mathbf{I}_m \right] \mathbf{D}(W_1, \dots, W_n) \left[[A]_1 \otimes \mathbf{I}_m, \dots, [A]_n \otimes \mathbf{I}_m \right]^\top \\
&= \left[[A]_1 \otimes W_1, \dots, [A]_n \otimes W_n \right] \left[[A]_1 \otimes \mathbf{I}_m, \dots, [A]_n \otimes \mathbf{I}_m \right]^\top \\
&= \sum_{j=1}^n \left([A]_j \otimes W_j \right) \left([A]_j^\top \otimes \mathbf{I}_m \right) = \sum_{j=1}^n [A]_j [A]_j^\top \otimes W_j,
\end{aligned}$$

and

$$\begin{aligned}
(A^\top \otimes \mathbf{I}_m)^\top W \mathbf{vec}(M) &= (A \otimes \mathbf{I}_m) \mathbf{D}(W_1, \dots, W_n) \mathbf{vec}(M) \\
&= (A \otimes \mathbf{I}_m) \mathbf{vec} \left(\left[W_1[M]_1, \dots, W_n[M]_n \right] \right) \\
&= \mathbf{vec} \left(\left[W_1[M]_1, \dots, W_n[M]_n \right] A^\top \right),
\end{aligned}$$

where in the last step we have applied the identity $(A^\top \otimes \mathbf{I}_m) \mathbf{vec}(B) = \mathbf{vec}(BA)$. ■

Finally, if W is a diagonal matrix, then $P^\top W P$ will also be a diagonal matrix, which we can further exploit.

Proposition 49. *A solution $\tilde{A}$ (respectively $\tilde{B}$) to the WLS problem $BA \approx M \in \mathbb{R}^{m \times n}$ with weight matrix $\mathbf{D}(\mathbf{vec}(W))$ for positive weights $W \in \mathbb{R}^{m \times n}$ corresponding element-wise to entries in M is given by solving the OLS problem*

$$\begin{aligned}
\mathbf{D}([W]_j)^{\frac{1}{2}} B [A]_j &\approx \mathbf{D}([W]_j)^{\frac{1}{2}} [M]_j && \text{for each column } [A]_j, \text{ or} \\
\mathbf{D}([W^\top]_i)^{\frac{1}{2}} A^\top [B^\top]_i &\approx \mathbf{D}([W^\top]_i)^{\frac{1}{2}} [M^\top]_i && \text{for each row } [B^\top]_i^\top \text{ of } B.
\end{aligned} \tag{10}$$

If the problem is overdetermined, a closed-form expression of the unique solution is given by solving the normal equations as

$$\begin{aligned}
[\tilde{A}]_j &= (B^\top \mathbf{D}([W]_j) B)^{-1} B^\top \mathbf{D}([W]_j) [M]_j && \text{for each column } [\tilde{A}]_j \text{ of } \tilde{A}, \text{ or} \\
[\tilde{B}^\top]_i &= (A \mathbf{D}([W^\top]_i) A^\top)^{-1} A \mathbf{D}([W^\top]_i) [M^\top]_i && \text{for each row } [\tilde{B}^\top]_i^\top \text{ of } \tilde{B}.
\end{aligned} \tag{11}$$

Proof. The first expression of Equation (10) is a special case of Equation (9), where $L_j^\top = \mathbf{D}([W]_j)^{\frac{1}{2}}$. The second expression follows from the first by observing that $\|M - BA\|_{\mathbf{D}(\mathbf{vec}(W))}^2 = \|M^\top - A^\top B^\top\|_{\mathbf{D}(\mathbf{vec}(W^\top))}^2$ and solving this for $B^\top$. ■

In the following Table 2 we summarize the computational complexity of finding the OLS, WLS, and GLS solutions to the overdetermined problem $BA \approx M \in \mathbb{R}^{m \times n}$

5.4 Selecting the Characterizer and Indicator

using the presented methods, i.e., by reducing the problems to OLS, which we then solve via the normal equations. Obtaining these complexity results is straight-forward if we note that the computational complexity of computing the matrix product AB for matrices $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{k \times n}$ is $\mathcal{O}(mkn)$, and the computational complexity of computing the Cholesky decomposition as well as the inverse of a symmetric positive-definite matrix $M \in \mathbb{R}^{n \times n}$ is $\mathcal{O}(n^3)$. Furthermore, we should point out that solving the GLS case with block-diagonal weight matrix W for B requires computing the Cholesky decomposition of a matrix of size $dm \times dm$, which imposes a strong memory requirement.

	solving for B	solving for A
OLS	$\mathcal{O}(dn(m+d))$	$\mathcal{O}(dm(n+d))$
WLS	$\mathcal{O}(d^2mn)$	$\mathcal{O}(d^2mn)$
GLS with block-diagonal W	$\mathcal{O}((dm)^2(dm+n))$	$\mathcal{O}(dm^2n)$
GLS	$\mathcal{O}((nm)^2dm)$	$\mathcal{O}((nm)^2dn)$

Table 2: The computational complexity of solving the overdetermined problem $BA \approx M$ with $B \in \mathbb{R}^{m \times d}$, $A \in \mathbb{R}^{d \times n}$, and $M \in \mathbb{R}^{m \times n}$ as an OLS, WLS, or GLS with according weights when solving either for B or for A using the methods presented above.

Now that we have examined various formulations of basic weighted linear regression, we turn to best weighted low-rank matrix approximations. We reformulate the task of finding the best rank- d approximation to a matrix in a way that includes weights and is amenable to an iterative optimization algorithm.

Definition 50. *The best weighted rank- d approximation to the matrix $M \in \mathbb{R}^{m \times n}$ with symmetric positive-definite weight matrix $W \in \mathbb{R}^{mn \times mn}$ is the matrix $\tilde{M}$ such that*

$$\tilde{M} = \operatorname{argmin}_{\tilde{M}} \left\{ \|M - \tilde{M}\|_W^2 : \operatorname{rank}(\tilde{M}) \leq d \right\},$$

or equivalently, $\tilde{M} = BA$, where

$$B, A = \operatorname{argmin}_{B, A} \left\{ \|M - BA\|_W^2 : B \in \mathbb{R}^{m \times d}, A \in \mathbb{R}^{d \times n} \right\}.$$

Note that for a given matrix M of rank greater or equal to d , the best (weighted) rank- d approximation $\tilde{M}$ must in fact have rank no less than d . The formulations are equivalent, since the matrix BA has rank at most d , while any matrix of rank at most d can be factored in this way. From the latter formulation of the problem we observe that the columns of B will form a basis of the column space of $\tilde{M}$, which we will still refer to as the *principal subspace* of M , while the columns of A will be the coordinate

5. LEARNING SEQUENTIAL SYSTEMS

representations of the columns of $\tilde{M}$ with respect to this basis B . This formulation also suggests an iterative approach to finding optimal B and A by successively optimizing B for a fixed A and vice versa, as done in Algorithm 7.

Algorithm 7: Best W -weighted rank- d approximation to M

In : $M \in \mathbb{R}^{m \times n}, W \in \mathbb{R}^{mn \times mn}, d$
Out: $B \in \mathbb{R}^{m \times d}, A \in \mathbb{R}^{d \times n}$, such that BA is the desired solution

- 1 $U_d, S_d, V_d^\top \leftarrow d$ -truncated SVD of M
 $B \leftarrow U_d S_d^{\frac{1}{2}}, A \leftarrow S_d^{\frac{1}{2}} V_d^\top$ // serves as an initial approximation
- 2 **repeat**
 - | solve the GLS $BA \approx M$ for B with weights W
 - | solve the GLS $BA \approx M$ for A with weights W
- until** convergence of $\|M - BA\|_W^2$

This algorithm is known as *alternating projections* and was described by Wentzell et al. [1997] for a latent variable formulation of PCA known as *maximum likelihood PCA*. This algorithm has been given the interpretation of an EM algorithm [Roweis, 1998], and it was shown for the unweighted case that this algorithm indeed converges to a global optimum [Tipping and Bishop, 1999]. The above method has linear convergence speed but upon convergence generally does give good results [Schuermans et al., 2005]. It is therefore sufficient for our purpose of investigating the usefulness of weights for small problem sizes. More specialized algorithms that achieve faster convergence do exist [Schuermans et al., 2005].

Note that the basis B for the principal subspace calculated in this way will generally not be orthonormal, but we do not require orthonormality. Furthermore, simpler structures of the weight matrix W , for instance a diagonal structure, enable the use of the simplifications for the GLS subproblems explained above.

Finally, we remark that the best W -weighted rank- d approximation $\tilde{M}$ to the matrix M is also a maximum likelihood estimator under the statistical model that $M = M_0 + E$ where $M_0 = BA$ has rank at most d and $E \sim \mathcal{N}(0, \lambda W^{-1})$ for some positive scalar λ , as can be seen analogously to Equation (6):

$$\mathcal{L}(BA | M) = \mathcal{N}(\text{vec}(M - BA) | 0, \lambda W^{-1}) = K \exp \left\{ -\frac{1}{2} \lambda^{-1} \|M - BA\|_W^2 \right\},$$

where K is a positive normalization constant.

5.4.3.2 How to Make Use of Weights

Now that we have introduced weighted versions of basic linear regression and low-rank matrix approximation, we are ready to substitute these for the currently unweighted regression tasks in the spectral learning algorithm.

Step (i):

We now estimate the column space of $\hat{F}_{Y,X}$ by computing the best $\mathbf{D}(\mathbf{vec}(W_{Y,X}))$ -weighted rank- d approximation $\tilde{F}_{Y,X} = \hat{B}\hat{A}$ to the matrix $\hat{F}_{Y,X}$. Using the diagonal weight matrix $\mathbf{D}(\mathbf{vec}(W_{Y,X}))$ means that we take the variance of each estimate in $\hat{F}_{Y,X}$ into account, but we disregard the covariance structure of the estimation errors. We thereby obtain a basis $\hat{B}$ for the estimated column space.

Step (ii):

Next we project $\hat{F}_Y^\top$ and the columns of $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ to the estimated principal subspace spanned by $\hat{B}$. Note that every such column is of the form $\hat{F}_{\bar{x}Y}^\top = [\hat{f}(\bar{x}y)]_{y \in Y}^\top$ for some word $\bar{x} \in \Sigma^*$. Instead of using an orthogonal projection, which maps a vector $\hat{F}_{\bar{x}Y}^\top$ to its closest vector in $\mathbf{span}\{\hat{B}\}$ measured by the Frobenius norm, which implicitly assumes the entries in $\hat{F}_{\bar{x}Y}^\top$ to have the same variance, we instead find the closest vector in $\mathbf{span}\{\hat{B}\}$ measured by the $\mathbf{D}(W_{\bar{x}Y}^\top)$ -weighted norm with weights $W_{\bar{x}Y}^\top$ for the estimates in $\hat{F}_{\bar{x}Y}^\top$, which takes the element-wise variance of the estimates in $\hat{F}_{\bar{x}Y}^\top$ into account. This means we need to find $\hat{A}_{\bar{x}Y}^\top$ that minimizes $\|\hat{F}_{\bar{x}Y}^\top - \hat{B}\hat{A}_{\bar{x}Y}^\top\|_{\mathbf{D}(W_{\bar{x}Y}^\top)}^2$, where $\hat{A}_{\bar{x}Y}^\top$ gives the coordinates of the desired projection $\hat{B}\hat{A}_{\bar{x}Y}^\top$ w.r.t. the basis $\hat{B}$.

This corresponds to solving the WLS $\hat{B}\hat{\omega}_\varepsilon \approx \hat{F}_Y^\top$ with weights $\mathbf{D}(W_Y^\top)$ for $\hat{\omega}_\varepsilon$, the WLS $\hat{B}\hat{A} \approx \hat{F}_{Y,X}$ with weights $\mathbf{D}(\mathbf{vec}(W_{Y,X}))$ for $\hat{A}$, and the WLS $\hat{B}\hat{A}_z \approx \hat{F}_{zY,X}$ with weights $\mathbf{D}(\mathbf{vec}(W_{zY,X}))$ for $\hat{A}_z$ for each $z \in \Sigma$, and thereby yields the required coordinate representations $\hat{\omega}_\varepsilon$, $\hat{A}$ and $\hat{A}_z$ w.r.t. the basis $\hat{B}$, for which we obtain the following expressions by Equation (11):

$$\begin{aligned}\hat{\omega}_\varepsilon &= (\hat{B}^\top \mathbf{D}(W_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(W_Y^\top) \hat{F}_Y^\top \\ [\hat{A}]_{\bar{x}} &= (\hat{B}^\top \mathbf{D}([W_{Y,X}]_{\bar{x}}) \hat{B})^{-1} \hat{B}^\top \mathbf{D}([W_{Y,X}]_{\bar{x}}) [\hat{F}_{Y,X}]_{\bar{x}} \\ [\hat{A}_z]_{\bar{x}} &= (\hat{B}^\top \mathbf{D}([W_{zY,X}]_{\bar{x}}) \hat{B})^{-1} \hat{B}^\top \mathbf{D}([W_{zY,X}]_{\bar{x}}) [\hat{F}_{zY,X}]_{\bar{x}}\end{aligned}\tag{12}$$

Note that $\hat{A}$ is already computed in Step (i) and we do not need to recompute it.

Step (iii):

We have obtained $\hat{A}$ and $\hat{A}_z$ that we can use as argument-value pairs for the estimation of $\hat{\tau}_z$, as well as $\hat{A}$ and $\hat{F}_X$ that we can use as argument-value pairs for the estimation of $\hat{\sigma}$. Instead of solving $\hat{\tau}_z \hat{A} \approx \hat{A}_z$ as well as $\hat{\sigma} \hat{A} \approx \hat{F}_X$ by OLS, we will now use GLS or

5. LEARNING SEQUENTIAL SYSTEMS

WLS with weights that should reflect the variances of the entries in $\hat{A}_z$ and $\hat{F}_X$, and in the case of GLS also take the covariances of the estimates in $\hat{A}_z$ into account. Note that all of these (weighted) least squares linear regression methods disregard the errors in $\hat{A}$. We still need to explain how to obtain appropriate weight matrices W_z for $\hat{A}_z$. For this, we require the following basic lemma that follows trivially from the linearity of the expectation.

Lemma 51. *Let $x = [x_1, \dots, x_n]^\top$ be a vector of random variables with covariance matrix $\mathbb{V}[x] = \mathbb{E}[(x - \mathbb{E}[x])(x - \mathbb{E}[x])^\top]$, and let $y = Tx$ for some $T \in \mathbb{R}^{m \times n}$. Then the covariance matrix $\mathbb{V}[y]$ of y is given by $\mathbb{V}[y] = T \mathbb{V}[x] T^\top$.*

From Equation (12) we see that every column $[\hat{A}_z]_{\bar{x}} = T_{\bar{x}}[\hat{F}_{zY,X}]_{\bar{x}}$ of $\hat{A}_z$ is computed in Step (ii) via a linear transformation $T_{\bar{x}}$ of the corresponding column of $\hat{F}_{zY,X}$, where $T_{\bar{x}} = (\hat{B}^\top \mathbf{D}([W_{zY,X}]_{\bar{x}}) \hat{B})^{-1} \hat{B}^\top \mathbf{D}([W_{zY,X}]_{\bar{x}})$. We consider $T_{\bar{x}}$ as given and invoke the above Lemma 51 to compute covariance matrix estimates $\hat{\mathbb{V}}[[\hat{A}_z]_{\bar{x}}]$ for each column indexed by the indicative word $\bar{x}$ from the covariance matrix estimates $\hat{\mathbb{V}}[[\hat{F}_{zY,X}]_{\bar{x}}] = \mathbf{D}([W_{zY,X}]_{\bar{x}})^{-1}$. We thereby obtain new weight matrices $W_{\bar{x}z} = \hat{\mathbb{V}}[[\hat{A}_z]_{\bar{x}}]^{-1}$ for each column of $\hat{A}_z$ indexed by $\bar{x}$ via

$$W_{\bar{x}z} = \hat{\mathbb{V}}[[\hat{A}_z]_{\bar{x}}]^{-1} = \left(T_{\bar{x}} \mathbf{D}([W_{zY,X}]_{\bar{x}})^{-1} T_{\bar{x}}^\top \right)^{-1} = \hat{B}^\top \mathbf{D}([W_{zY,X}]_{\bar{x}}) \hat{B},$$

which is clearly a symmetric positive-definite matrix. Since we have assumed the entries in $\hat{F}_{zY,X}$ to be uncorrelated, we can assume uncorrelated columns for $\hat{A}_z$ and can therefore assemble the weight matrices $W_{\bar{x}z}$ into a block-diagonal weight matrix W_z for $\hat{A}_z$ as

$$W_z = \mathbf{D}(W_{\bar{x}_1 z}, \dots, W_{\bar{x}_{|X|} z}), \quad \text{where } X = \{\bar{x}_1, \dots, \bar{x}_{|X|}\}.$$

In summary, the above discussion yields the weighted spectral learning Algorithm 8. Note that we can achieve significant computational savings in Step (iii) if we only make use of the element-wise variances of the entries in $\hat{A}_z$ but disregard their covariances, i.e., if we set all off-diagonal entries in W_z to zero, since we then compute the matrices $\hat{\tau}_z$ by solving a WLS problem instead of a GLS problem. We will refer to this simplified version as the *element-wise weighted* spectral learning algorithm.

At this point we should examine how the choice of basis $\hat{B}$ for the estimated principal subspace, which is estimated in Step (i) as the column space of the best weighted rank- d approximation to $\hat{F}_{Y,X}$, affects the model estimated by the weighted spectral learning Algorithm 8. The following Lemma 52 gives the expected answer, namely, that a different choice of basis will indeed lead to an equivalent model estimate.

5.4 Selecting the Characterizer and Indicator

Algorithm 8: [Element-wise] Weighted Spectral Learning

In : Estimates $\hat{F}_{Y,X}$, $\hat{F}_{zY,X}$, $\hat{F}_Y^\top$, and $\hat{F}_X$ and corresponding weight matrices $W_{Y,X}$, $W_{zY,X}$, $W_Y^\top$, and W_X of element-wise weights $w(\bar{x}) = \hat{\mathbb{V}}_{2,3}[\hat{f}(\bar{x})]^{-1}$, the model dimension d

Out: An estimated d -dimensional SS model $\hat{\mathcal{M}} = (\hat{\sigma}, \{\hat{\tau}_z\}, \hat{\omega}_\varepsilon)$

- 1 $\hat{B}\hat{A} \leftarrow$ best $\mathbf{D}(\mathbf{vec}(W_{Y,X}))$ -weighted rank- d approximation to $\hat{F}_{Y,X}$
 - 2 **for** $z \in \Sigma$ **do**
 - $\hat{A}_z \leftarrow$ WLS solution to $\hat{B}\hat{A}_z \approx \hat{F}_{zY,X}$ with weights $\mathbf{D}(\mathbf{vec}(W_{zY,X}))$
 - $\hat{\omega}_\varepsilon \leftarrow$ WLS solution to $\hat{B}\hat{\omega}_\varepsilon \approx \hat{F}_Y^\top$ with weights $\mathbf{D}(W_Y^\top)$
 - 3 **for** $z \in \Sigma$ **do**
 - for** $\bar{x} \in X$ **do**
 - $W_{\bar{x}z} \leftarrow \hat{B}^\top \mathbf{D}([W_{zY,X}]_{\bar{x}}) \hat{B}$
 - $W_z \leftarrow \mathbf{D}(W_{\bar{x}_1 z}, \dots, W_{\bar{x}_{|X|} z})$
 - if** *element-wise weighted version* **then**
 - set all off-diagonal entries in W_z to zero
 - $\hat{\tau}_z \leftarrow$ GLS (or WLS) solution to $\hat{\tau}_z \hat{A} \approx \hat{A}_z$ with weights W_z
 - $\hat{\sigma} \leftarrow$ WLS solution to $\hat{\sigma} \hat{A} \approx \hat{F}_X$ with weights $\mathbf{D}(W_X)$
-

Lemma 52. Let $\hat{B}'$ be a different basis for the column space of the best weighted rank- d approximation $\tilde{F}_{Y,X} = \hat{B}\hat{A}$ to $\hat{F}_{Y,X}$ computed in Step (i) of Algorithm 8. Then the model estimate $\mathcal{M}'$ computed via Algorithm 8 with this choice of basis $\hat{B}'$ is equivalent to the model estimate $\mathcal{M}$ computed using the original choice of basis $\hat{B}$.

Proof. We have $\hat{B}'\hat{A}' = \hat{B}\hat{A}$, which means that $\hat{B}' = \hat{B}\rho^{-1}$ for some non-singular $\rho \in \mathbb{R}^{d \times d}$. Step (ii) then computes $\hat{A}' = \rho\hat{A}$, $\hat{A}'_z = \rho\hat{A}_z$, and $\hat{\omega}'_\varepsilon = \rho\hat{\omega}_\varepsilon$ by Equation (12). For Step (iii) we first compute $W'_{\bar{x}z} = (\rho^{-1})^\top W_{\bar{x}z} \rho^{-1} = (\rho^{-1})^\top L_{\bar{x}} L_{\bar{x}}^\top \rho^{-1}$, where $L_{\bar{x}} L_{\bar{x}}^\top$ is the Cholesky decomposition of $W_{\bar{x}z}$, and then need to minimize

$$\begin{aligned}
\|\hat{A}'_z - \hat{\tau}'_z \hat{A}'\|_{W'_z} &= \|\rho\hat{A}_z - \hat{\tau}'_z \rho\hat{A}\|_{\mathbf{D}(W'_{\bar{x}_1 z}, \dots, W'_{\bar{x}_{|X|} z})} \\
&= \sum_{\bar{x} \in X} \|L_{\bar{x}}^\top \rho^{-1} (\rho[\hat{A}_z]_{\bar{x}} - \hat{\tau}'_z \rho[\hat{A}]_{\bar{x}})\|_F \\
&= \sum_{\bar{x} \in X} \|L_{\bar{x}}^\top ([\hat{A}_z]_{\bar{x}} - \rho^{-1} \hat{\tau}'_z \rho[\hat{A}]_{\bar{x}})\|_F \\
&= \|\hat{A}_z - (\rho^{-1} \hat{\tau}'_z \rho) \hat{A}\|_{W_z},
\end{aligned}$$

where we have made use of the block-diagonal structure of W_z and W'_z , which means columns are weighted independently. We see that this is solved by $\hat{\tau}'_z = \rho \hat{\tau}_z \rho^{-1}$, since

5. LEARNING SEQUENTIAL SYSTEMS

$\hat{\tau}_z$ minimizes $\|\hat{A}_z - \hat{\tau}_z \hat{A}\|_{W_z}$. Furthermore, the $D(W_X)$ -weighted WLS $\hat{\sigma}' \rho \hat{A} \approx \hat{F}_X$ is clearly solved by $\hat{\sigma}' = \hat{\sigma} \rho^{-1}$. This shows that the estimated models will be conjugated by $\mathcal{M}' = \rho \mathcal{M} \rho^{-1}$. ■

Let us also examine the computational complexity of the weighted spectral learning Algorithm 8.

Proposition 53. *The computational complexity of the weighted spectral learning Algorithm 8 is $\mathcal{O}(d^2|X||Y|(|\Sigma| + K))$ in the case of the element-wise weighted variant and $\mathcal{O}(d^2|X||Y|(|\Sigma| + K) + |\Sigma|d^4(d^2 + |X|))$ in the general case, where $|\Sigma|$ is the alphabet size, d is the target dimension, $|X|$ and $|Y|$ are the sizes of the sets of indicative and characteristic words, and K is the number of iterations required for the Algorithm 7 that is used in Step (i) to converge.*

Proof. Using the complexities listed in Table 2, we see that Algorithm 7 used in the first step requires computing a d -truncated SVD as well as solving $2K$ WLS problems, while Step (ii) requires solving $|\Sigma|$ WLS problems of the same size, which together takes time in $\mathcal{O}(d^2|X||Y|(K + |\Sigma|))$. Computing W_z in Step (iii) requires time in $\mathcal{O}(d^2|X||Y|)$ for each symbol $z \in \Sigma$. The computation of the estimates $\hat{\tau}_z$ in Step (iii) requires $\mathcal{O}(d^3|X|)$ operations for every symbol $z \in \Sigma$ to solve the WLS problems in the element-wise weighted case and $\mathcal{O}(d^4(d^2 + |X|))$ operations for every symbol $z \in \Sigma$ to solve the GLS problems in the general case. The WLS problems that need to be solved for $\hat{\omega}_\varepsilon$ and $\hat{\sigma}$ are smaller. ■

Finally, we should remark that exploiting the quality of the estimates $\hat{f}(\bar{x})$ by estimating their variances and incorporating weights into the spectral learning algorithm is still not making use of the rich structure of the matrix F . Furthermore, our current approach using WLS or GLS for the final step of the weighted spectral learning algorithm to solve $\hat{\tau}_z \hat{A} = \hat{A}_z$ only accounts for errors in the matrix $\hat{A}_z$, but not in the matrix $\hat{A}$. Methods that take errors both in $\hat{A}$ as well as in $\hat{A}_z$ into account exist and are known as total least squares [Markovsky and Van Huffel, 2007b].

5.4.4 A Row and Column Weighted Spectral Algorithm

We now introduce a simplified version of the above weighted spectral algorithm that uses only row-wise and column-wise weighting. This results in a non-iterative and very simple procedure akin to the standard spectral learning Algorithm 6 that, however, still retains the core idea of taking the precision of estimates into account. As with the spectral learning Algorithm 6, the row and column weighted spectral algorithm

5.4 Selecting the Characterizer and Indicator

we describe here leads to a particular selection of characterizer and indicator matrices C and Q for use in the learning Equations (2). Furthermore, we will see in the later Section 5.4.6 that this row and column weighted spectral algorithm is, in fact, closely related to the ES algorithm.

More precisely, we will use a single weight to measure the overall precision of all the estimates in a row or column of the matrix $\hat{F}_{Y,X}$ (and similarly for $\hat{F}_{zY,X}$). The weight for an entry is then given by the product of the column and the row weight, i.e., we assume the weight matrix $W_{Y,X}$ can be approximately factored as $W_{Y,X} \approx w_Y^\top w_X$ for an appropriate vector $w_Y^\top$ of row weights and row vector w_X of column weights. Furthermore, due to the similar makeup of the matrices $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$, we will assume that the weight matrices $W_{Y,X}$ and $W_{zY,X}$ differ only by a scalar factor α_z , i.e., that $W_{zY,X} \approx \alpha_z W_{Y,X}$, and finally we assume that in fact $w_Y^\top \approx \alpha W_Y^\top$ and $w_X \approx \tilde{\alpha} W_X$, i.e., that the weight for a row or column indexed by a word $\bar{x}$ is approximately given by the weight $w(\bar{x})$ for the estimate $\hat{f}(\bar{x})$, at least up to a scalar factor α or $\tilde{\alpha}$. We can summarize the restrictions on the weights that we impose as follows:

$$W_{Y,X} \propto W_{zY,X} \propto w_Y^\top w_X \quad \text{with} \quad w_Y^\top \propto W_Y^\top \quad \text{and} \quad w_X \propto W_X.$$

Next, we discuss how to obtain suitable row and column weight vectors $w_Y^\top$ and w_X from the available element-wise weights. First note that the scalar factors will have no effect on the model estimated by the weighted spectral learning Algorithm 8, so we may ignore them. A reasonable choice may therefore be to simply use $w_Y^\top = W_Y^\top$ as row weights and $w_X = W_X$ as column weights. Alternatively, we may argue that the row (or column) weights should reflect the average variance (and therefore the expected mean squared error) of the corresponding row (or column) of $W_{Y,X}$, which gives

$$\begin{aligned} w_Y^\top &= \left[\left(\frac{\sum_{\bar{x} \in X} [W_{Y,X}]_{\bar{y},\bar{x}}^{-1}}{|X|} \right)^{-1} \right]_{\bar{y} \in Y}^\top = \left[\left(\frac{\sum_{\bar{x} \in X} \hat{V}[\hat{f}(\bar{x}\bar{y})]}{|X|} \right)^{-1} \right]_{\bar{y} \in Y}^\top \\ w_X &= \left[\left(\frac{\sum_{\bar{y} \in Y} [W_{Y,X}]_{\bar{y},\bar{x}}^{-1}}{|Y|} \right)^{-1} \right]_{\bar{x} \in X} = \left[\left(\frac{\sum_{\bar{y} \in Y} \hat{V}[\hat{f}(\bar{x}\bar{y})]}{|Y|} \right)^{-1} \right]_{\bar{x} \in X} \end{aligned} \quad (13)$$

Note that if the matrix $W_{Y,X}$ of element-wise weights can be factored exactly as $W_{Y,X} = \tilde{w}_Y^\top \tilde{w}_X$ for some vectors $\tilde{w}_Y^\top$ and $\tilde{w}_X$, then we can easily check that the calcu-

5. LEARNING SEQUENTIAL SYSTEMS

lation from Equation (13) gives

$$w_Y^\top = \tilde{w}_Y^\top \left(\frac{\sum_{\tilde{x} \in X} [\tilde{w}_X]_{\tilde{x}}^{-1}}{|X|} \right)^{-1} \quad \text{and} \quad w_X = \tilde{w}_X \left(\frac{\sum_{\tilde{y} \in Y} [\tilde{w}_Y]_{\tilde{y}}^{-1}}{|Y|} \right)^{-1},$$

which means that the factorization of $W_{Y,X}$ is recovered up to scalar factors that can be safely disregarded. We will prefer this approach – computing row and column weights $w_Y^\top$ and w_X by averaging from the element-wise weight matrix $W_{Y,X}$ – to the simpler approach of just using the weight vectors $w_Y^\top = W_Y^\top$ and $w_X = W_X$. Furthermore, due to the averaging, it is less important to enforce a lower bound on the element-wise variance estimates, which introduces bias, and we will therefore use the variance estimates $\mathbb{V}_{2,0}[\hat{f}(\tilde{x})]$ to derive row and column weights. This leads to the same row and column weighting that we have employed as a normalization step when computing the numerical rank in Section 5.3.1.

Now let us assume that we have obtained suitable row and column weight vectors $w_Y^\top$ and w_X and use these to compute element-wise weights via $W_{Y,X} = W_{zY,X} = w_Y^\top w_X$ and $W_Y^\top = w_Y^\top$ as well as $W_X = w_X$ for use with the weighted spectral learning Algorithm 8. We will use the following Lemma 54 to gain significant computational simplifications.

Lemma 54. *Let $M \in \mathbb{R}^{m \times n}$, and let $W \in \mathbb{R}^{mn \times mn}$ be a weight matrix that can be factored as $W = W_c \otimes W_r$ for symmetric and positive-definite matrices $W_r \in \mathbb{R}^{m \times m}$ and $W_c \in \mathbb{R}^{n \times n}$ with decompositions $W_r = L_r L_r^\top$ and $W_c = L_c L_c^\top$, e.g., their Cholesky decompositions. Then $\|M\|_W^2 = \|L_r^\top M L_c\|_F^2$.*

In the special case of a weight matrix $W = \mathbf{D}(\text{vec}(w_Y^\top w_X))$ for a vector $w_Y^\top \in \mathbb{R}^m$ of positive row weights and a row vector $w_X \in \mathbb{R}^n$ of positive column weights this gives $\|M\|_{\mathbf{D}(\text{vec}(w_Y^\top w_X))}^2 = \|\mathbf{D}(w_Y^\top)^{\frac{1}{2}} M \mathbf{D}(w_X)^{\frac{1}{2}}\|_F^2$.

Proof. First note that $L = L_c \otimes L_r$ gives a decomposition of $W = W_c \otimes W_r$, since $LL^\top = (L_c \otimes L_r)(L_c \otimes L_r)^\top = L_c L_c^\top \otimes L_r L_r^\top = W_c \otimes W_r = W$, and W will also be symmetric and positive-definite. Therefore, $\|M\|_W^2 = \|(L_c \otimes L_r)^\top \text{vec}(M)\|_F^2 = \|\text{vec}(L_r^\top M L_c)\|_F^2 = \|L_r^\top M L_c\|_F^2$, where we have used the identity $(B^\top \otimes A) \text{vec}(C) = \text{vec}(ACB)$ for matrices A, B, C .

For the special case of $W = \mathbf{D}(\text{vec}(w_Y^\top w_X)) = \mathbf{D}(w_X^\top \otimes w_Y^\top) = \mathbf{D}(w_X) \otimes \mathbf{D}(w_Y^\top)$, we may use $W_r = \mathbf{D}(w_Y^\top)$ and $W_c = \mathbf{D}(w_X)$ with decompositions $L_r = \mathbf{D}(w_Y^\top)^{\frac{1}{2}}$ and $L_c = \mathbf{D}(w_X)^{\frac{1}{2}}$. ■

5.4 Selecting the Characterizer and Indicator

Step (i):

For the first step of the weighted spectral learning algorithm we need to compute the best $\mathbf{D}(\mathbf{vec}(W_{Y,X}))$ -weighted rank- d approximation to the matrix $\hat{F}_{Y,X}$, which means we need to find a rank- d matrix $\hat{B}\hat{A}$ that minimizes

$$\begin{aligned}\|\hat{F}_{Y,X} - \hat{B}\hat{A}\|_{\mathbf{D}(\mathbf{vec}(W_{Y,X}))}^2 &= \|\mathbf{D}(w_Y^\top)^{\frac{1}{2}}(\hat{F}_{Y,X} - \hat{B}\hat{A})\mathbf{D}(w_X)^{\frac{1}{2}}\|_{\mathbf{F}}^2 \\ &= \|\mathbf{D}(w_Y^\top)^{\frac{1}{2}}\hat{F}_{Y,X}\mathbf{D}(w_X)^{\frac{1}{2}} - \mathbf{D}(w_Y^\top)^{\frac{1}{2}}\hat{B}\hat{A}\mathbf{D}(w_X)^{\frac{1}{2}}\|_{\mathbf{F}}^2.\end{aligned}$$

From this we observe that $\mathbf{D}(w_Y^\top)^{\frac{1}{2}}\hat{B}\hat{A}\mathbf{D}(w_X)^{\frac{1}{2}}$ is given by the best rank- d approximation to $\mathbf{D}(w_Y^\top)^{\frac{1}{2}}\hat{F}_{Y,X}\mathbf{D}(w_X)^{\frac{1}{2}}$, which is given by the d -truncated SVD $U_d S_d V_d^\top$ of $\mathbf{D}(w_Y^\top)^{\frac{1}{2}}\hat{F}_{Y,X}\mathbf{D}(w_X)^{\frac{1}{2}}$ as per the Eckart-Young theorem. Then $\mathbf{D}(w_Y^\top)^{\frac{1}{2}}\hat{B}\hat{A}\mathbf{D}(w_X)^{\frac{1}{2}} = U_d S_d V_d^\top$ and we may use $\hat{B} = \mathbf{D}(w_Y^\top)^{-\frac{1}{2}}U_d$ and $\hat{A} = S_d V_d^\top \mathbf{D}(w_X)^{-\frac{1}{2}}$.

Step (ii):

Here we simply follow Step (ii) of the weighted spectral learning from the previous Section 5.4.3, but use that $W_Y^\top = w_Y$ and $[W_{Y,X}]_{\bar{x}} = [W_{zY,X}]_{\bar{x}} = w_Y^\top[w_X]_{\bar{x}}$, which means the expressions given in Equation (12) simplify to

$$\begin{aligned}\hat{\omega}_\varepsilon &= (\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{F}_Y^\top, \\ \hat{A} &= (\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{F}_{Y,X}, \\ \hat{A}_z &= (\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{F}_{zY,X}.\end{aligned}$$

Step (iii):

First we need to compute weights W_z for $\hat{A}_z$. Noting that $[W_{zY,X}]_{\bar{x}} = w_Y^\top[w_X]_{\bar{x}}$ we compute $W_{\bar{x}z} = \hat{B}^\top \mathbf{D}([W_{zY,X}]_{\bar{x}}) \hat{B} = [w_X]_{\bar{x}} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B}$, and $W_z = \mathbf{D}(W_{\bar{x}1z}, \dots, W_{\bar{x}|X|z}) = \mathbf{D}(w_X) \otimes \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B}$. Let $W_r = \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B}$, and note that W_r is symmetric and positive-definite, so it has a Cholesky decomposition $W_r = L_r L_r^\top$. Solving the W_z -weighted GLS $\hat{\tau}_z \hat{A} \approx \hat{A}_z$ then means finding the $\hat{\tau}_z$ that minimizes $\|\hat{\tau}_z \hat{A} - \hat{A}_z\|_{\mathbf{D}(w_X) \otimes W_r}^2 = \|L_r^\top \hat{\tau}_z \hat{A} \mathbf{D}(w_X)^{\frac{1}{2}} - L_r^\top \hat{A}_z \mathbf{D}(w_X)^{\frac{1}{2}}\|_{\mathbf{F}}^2$, which requires solving the OLS $L_r^\top \hat{\tau}_z \hat{A} \mathbf{D}(w_X)^{\frac{1}{2}} \approx L_r^\top \hat{A}_z \mathbf{D}(w_X)^{\frac{1}{2}}$, a solution of which can be expressed by the normal equation $L_r^\top \hat{\tau}_z = L_r^\top \hat{A}_z \mathbf{D}(w_X) \hat{A}^\top (\hat{A} \mathbf{D}(w_X) \hat{A}^\top)^{-1}$, which gives

$$\begin{aligned}\hat{\tau}_z &= \hat{A}_z \mathbf{D}(w_X) \hat{A}^\top (\hat{A} \mathbf{D}(w_X) \hat{A}^\top)^{-1} \\ &= \hat{A}_z \mathbf{D}(w_X)^{\frac{1}{2}} (\hat{A} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger \quad \left[\text{note } (\hat{A} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger = \mathbf{D}(w_X)^{\frac{1}{2}} \hat{A}^\top (\hat{A} \mathbf{D}(w_X) \hat{A}^\top)^{-1} \right] \\ &= (\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{F}_{zY,X} \mathbf{D}(w_X)^{\frac{1}{2}} \\ &\quad \cdot ((\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{F}_{Y,X} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger.\end{aligned}$$

5. LEARNING SEQUENTIAL SYSTEMS

To obtain $\hat{\sigma}$, we need to solve the $\mathbf{D}(w_X)$ -weighted WLS $\hat{\sigma}\hat{A} \approx \hat{F}_X$, which by Equation (11) is given by $\hat{\sigma} = \hat{F}_X \mathbf{D}(w_X) \hat{A}^\top (\hat{A} \mathbf{D}(w_X) \hat{A}^\top)^{-1} = \hat{F}_X \mathbf{D}(w_X)^{\frac{1}{2}} (\hat{A} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger$, or $\hat{\sigma} = \hat{F}_X \mathbf{D}(w_X)^{\frac{1}{2}} ((\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{F}_{Y,X} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger$.

If we inspect the formulas for the computed $(\hat{\sigma}, \{\hat{\tau}_z\}, \hat{\omega}_\varepsilon)$ we see that this is exactly what the learning Equations (2) give if we use the characterizer and indicator

$$\begin{aligned} C &= (\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \\ Q &= \mathbf{D}(w_X)^{\frac{1}{2}} ((\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \hat{F}_{Y,X} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger \\ &= \mathbf{D}(w_X)^{\frac{1}{2}} (C \hat{F}_{Y,X} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger, \end{aligned} \quad (14)$$

since we can easily see that $C \hat{F}_{Y,X} Q = \mathbf{I}_d$. Now, substituting the basis $\hat{B} = \mathbf{D}(w_Y^\top)^{-\frac{1}{2}} U_d$ obtained in Step (i), we get $\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B} = U_d^\top U_d = \mathbf{I}_d$ and hence $C = U_d^\top \mathbf{D}(w_Y^\top)^{\frac{1}{2}}$ and $Q = \mathbf{D}(w_X)^{\frac{1}{2}} (U_d^\top \mathbf{D}(w_Y^\top)^{\frac{1}{2}} \hat{F}_{Y,X} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger = \mathbf{D}(w_X)^{\frac{1}{2}} (U_d^\top U S V^\top)^\dagger = \mathbf{D}(w_X)^{\frac{1}{2}} V_d S_d^{-1}$, since $U_d^\top U S V^\top = [\mathbf{I}_d, 0] S V^\top = S_d V_d^\top$ and $(S_d V_d^\top)^\dagger = V_d S_d (S_d V_d^\top V_d S_d)^{-1} = V_d S_d^{-1}$. We obtain the following simple modification to the spectral learning Algorithm 6, which clearly has the same computational complexity of $\mathcal{O}(d|X||Y|)$.

Algorithm 9: Row and column weighted spectral learning

In : Estimates $\hat{F}_{Y,X}$ and a corresponding weight matrix $W_{Y,X}$ with element-wise weights $w(\bar{x}) = \hat{\mathbb{V}}_{2,0}[\hat{f}(\bar{x})]^{-1}$; the model dimension d
Out: C, Q for use in the learning Equations (2)

$$\begin{aligned} w_Y^\top &\leftarrow \left[\left(\frac{\sum_{\bar{x} \in X} [W_{Y,X}]_{\bar{y}, \bar{x}}^{-1}}{|X|} \right)^{-1} \right]_{\bar{y} \in Y}^\top, \quad w_X \leftarrow \left[\left(\frac{\sum_{\bar{y} \in Y} [W_{Y,X}]_{\bar{y}, \bar{x}}^{-1}}{|Y|} \right)^{-1} \right]_{\bar{x} \in X} \\ U_d, S_d, V_d^\top &\leftarrow d\text{-truncated SVD of } \mathbf{D}(w_Y^\top)^{\frac{1}{2}} \hat{F}_{Y,X} \mathbf{D}(w_X)^{\frac{1}{2}} \\ C &\leftarrow U_d^\top \mathbf{D}(w_Y^\top)^{\frac{1}{2}}, \quad Q \leftarrow \mathbf{D}(w_X)^{\frac{1}{2}} V_d S_d^{-1} \end{aligned}$$

5.4.5 The Error Controlling Algorithm

The error controlling (EC) approach of Zhao et al. [2009a,b] selects characterizer and indicator matrices C and Q that minimize an error bound for the *relative approximation error* of the estimated model parameters as given in Proposition 55. This algorithm was originally formulated for OOMs only and made use of the normalization $\sigma = \mathbb{1}_d^\top$ that is often used in the context of OOMs. This in turn imposed additional restrictions on the admissible selections of indicative and characteristic words. Here, we present

5.4 Selecting the Characterizer and Indicator

a more general and yet simplified EC approach that eliminates these restrictions and applies to learning SMA, OOMs, IO-OOMs, and PSRs alike. Additionally, we show that the EC algorithm is equivalent to the spectral learning Algorithm 6.

Proposition 55. *Let $f : \Sigma \rightarrow \mathbb{R}$ be a stochastic language or (controlled) stochastic process of rank d , let X and Y be sets of indicative and characteristic words such that $\text{rank}(F_{Y,X}) = d$ and let $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ be estimates for $F_{Y,X}$ and $F_{zY,X}$. Let $C \in \mathbb{R}^{d \times |Y|}$ and $Q \in \mathbb{R}^{|X| \times d}$ be any choice of characterizer and indicator such that $CF_{Y,X}Q$ as well as $C\hat{F}_{Y,X}Q$ are invertible. Let τ_z and $\hat{\tau}_z$ be computed from $F_{Y,X}$ and $F_{zY,X}$ or, respectively, $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ via the learning Equations (2), and define $\tau_* = [\tau_z^\top]_{z \in \Sigma}^\top$ and $\hat{\tau}_* = [\hat{\tau}_z^\top]_{z \in \Sigma}^\top$ by vertically stacking the τ_z or $\hat{\tau}_z$ operators, as well as $F_* = [F_{zY,X}^\top]_{z \in \Sigma}^\top$ and $\hat{F}_* = [\hat{F}_{zY,X}^\top]_{z \in \Sigma}^\top$ by vertically stacking the matrices $F_{zY,X}$ or $\hat{F}_{zY,X}$. Let $E_{Y,X} = F_{Y,X} - \hat{F}_{Y,X}$ and $E_* = F_* - \hat{F}_*$.*

Then the relative approximation error of $\hat{\tau}_*$ satisfies the bound

$$\frac{\|\tau_* - \hat{\tau}_*\|_F}{\|\tau_*\|_F} \leq \|C\|_F \|Q(C\hat{F}_{Y,X}Q)^{-1}\|_F \left(\sqrt{\frac{|\Sigma|}{\sum_{z \in \Sigma} \rho(\tau_z)^2}} \|E_*\|_F + \|E\|_F \right),$$

where the quantity $\kappa = \|C\|_F \|Q(C\hat{F}_{Y,X}Q)^{-1}\|_F$ is called the robustness indicator and $\rho(\tau_z)$ is the spectral radius of the matrix τ_z . Furthermore, the quantities $\rho(\tau_z)$ for $z \in \Sigma$ are independent of the choice of C and Q .

Proof. [adapted from Zhao et al., 2009a] Let $C_* = \mathbf{D}(C, \dots, C)$ ($|\Sigma|$ copies of C). Using the introduced notation the learning Equations (2) can be written concisely to obtain:

$$\begin{aligned} \tau_* &= (C_* F_* Q)(C F_{Y,X} Q)^{-1} \\ &= (C_* \hat{F}_* Q + C_* E_* Q)(C \hat{F}_{Y,X} Q + C E Q)^{-1} \\ &= (C_* \hat{F}_* Q + C_* E_* Q)(C \hat{F}_{Y,X} Q)^{-1} [\mathbf{I}_d + C E Q (C \hat{F}_{Y,X} Q)^{-1}]^{-1} \\ &= [\hat{\tau}_* + (C_* E_* Q)(C \hat{F}_{Y,X} Q)^{-1}] [\mathbf{I}_d + C E Q (C \hat{F}_{Y,X} Q)^{-1}]^{-1}, \end{aligned}$$

which implies $\tau_* + \tau_* C E Q (C \hat{F}_{Y,X} Q)^{-1} = \hat{\tau}_* + (C_* E_* Q)(C \hat{F}_{Y,X} Q)^{-1}$. By rearranging, taking Frobenius norms, and using the triangle inequality and submultiplicativity,

$$\begin{aligned} \|\tau_* - \hat{\tau}_*\|_F &= \|(C_* E_* - \tau_* C E) Q (C \hat{F}_{Y,X} Q)^{-1}\|_F \\ &\leq (\|C_*\|_F \|E_*\|_F + \|\tau_*\|_F \|C\|_F \|E\|_F) \|Q(C \hat{F}_{Y,X} Q)^{-1}\|_F \\ &= \|\tau_*\|_F \|C\|_F \|Q(C \hat{F}_{Y,X} Q)^{-1}\|_F \left(\frac{\|C_*\|_F}{\|C\|_F} \|\tau_*\|_F^{-1} \|E_*\|_F + \|E\|_F \right). \end{aligned}$$

5. LEARNING SEQUENTIAL SYSTEMS

Now $\frac{\|C_*\|_F}{\|C\|_F} = \sqrt{|\Sigma|}$, and $\|\tau_*\|_F^2 = \sum_{z \in \Sigma} \|\tau_z\|_F^2 \geq \sum_{z \in \Sigma} \|\tau_z\|_2^2 \geq \sum_{z \in \Sigma} \rho(\tau_z)^2$, and the claimed bound follows.

A different choice of characterizer C' and indicator Q' satisfying the requirement that $C'F_{Y,X}Q' \in \mathbb{R}^{d \times d}$ is invertible will lead to operators τ'_z of an equivalent (minimal) model by Proposition 35, which therefore must be related to the operators τ_z by $\tau'_z = T\tau_zT^{-1}$ for some non-singular transformation T according to Proposition 14. Then $\rho(\tau'_z) = \rho(T\tau_zT^{-1}) = \rho(\tau_z)$ for all $z \in \Sigma$, which shows that the quantities $\rho(\tau_z)$ are indeed independent of the choice of C and Q . ■

Note that in the original publication the inequality $\|\tau_*\|_F \geq \frac{1}{\sqrt{|\Sigma|}}$ was used instead, which depended on the columns of τ_* summing to 1. This was in turn ensured by adding additional restrictions on the choice of characteristic words and characterizer C . These restrictions are now no longer needed.

In practice, we cannot guarantee $d = \text{rank}(F_{Y,X}) = \text{rank}(F)$. Instead, we select indicative and characteristic words X and Y as well as a target dimension d in some reasonable way, as we have already discussed in previous sections, and simply assume these conditions to be met. Similarly, we cannot guarantee $CF_{Y,X}Q$ to be invertible. However, the selection of C and Q will be further restricted such that $C\hat{F}_{Y,X}Q = \mathbf{I}_d$, so it is reasonable to assume that $CF_{Y,X}Q \approx C\hat{F}_{Y,X}Q = \mathbf{I}_d$ will in fact be invertible.

The EC algorithm selects C, Q , subject to $C\hat{F}_{Y,X}Q$ being invertible, that minimize the bound on the relative approximation error from Proposition 55, which means C and Q must minimize the robustness indicator $\kappa = \|C\|_F \|Q(C\hat{F}_{Y,X}Q)^{-1}\|_F$, as only κ depends on C and Q . This is equivalent to the optimization problem

$$(C, Q) = \arg \min_{(C, Q)} \{\|C\|_F \|Q\|_F : C\hat{F}_{Y,X}Q = \mathbf{I}_d\}, \quad (15)$$

which uses the stronger restriction $C\hat{F}_{Y,X}Q = \mathbf{I}_d$ on the choice of C and Q . To see that we may indeed use the requirement $C\hat{F}_{Y,X}Q = \mathbf{I}_d$ without loss of generality, note that for any pair (C, Q) that minimizes κ subject to $C\hat{F}_{Y,X}Q$ being invertible we get a pair (C, Q') by setting $Q' = Q(C\hat{F}_{Y,X}Q)^{-1}$ that attains the same minimum value $\|C\|_F \|Q(C\hat{F}_{Y,X}Q)^{-1}\|_F = \|C\|_F \|Q'\|_F$ but satisfies $(C\hat{F}_{Y,X}Q') = \mathbf{I}_d$.

The optimization problem from Equation (15) can be solved efficiently by the iterative procedure shown in Algorithm 10 [Zhao et al., 2009a] in time $\mathcal{O}(d|X||Y|K)$, where K is the required number of iterations.

Although not previously realized, Algorithm 10 turns out to be essentially the *alternating projections* Algorithm 7 that is used to compute the best (unweighted)

5.4 Selecting the Characterizer and Indicator

Algorithm 10: The Error Controlling (EC) algorithm

In : Estimates $\hat{F}_{Y,X}$; the model dimension d
Out: C, Q for use in the learning Equations (2)
initialize $C \in \mathbb{R}^{d \times |Y|}$ and $Q \in \mathbb{R}^{|X| \times d}$ randomly
repeat
| $C \leftarrow (\hat{F}_{Y,X}Q)^\dagger, \quad Q \leftarrow (C\hat{F}_{Y,X})^\dagger$
until convergence of $\|C\|_F\|Q\|_F$

rank- d approximation BA to the matrix $\hat{F}_{Y,X}$ via the iteration $B \leftarrow \hat{F}_{Y,X}A^\dagger$ and $A \leftarrow B^\dagger\hat{F}_{Y,X}$ (solving the unweighted GLS, i.e., OLS, $BA \approx \hat{F}_{Y,X}$ for B and then A , respectively). To see the relationship, note that B is updated from one iteration to the next via $B \leftarrow \hat{F}_{Y,X}(B^\dagger\hat{F}_{Y,X})^\dagger$, while C is updated from one iteration to the next by $C \leftarrow (\hat{F}_{Y,X}(C\hat{F}_{Y,X})^\dagger)^\dagger$. Therefore, if we set $C = B^\dagger$ and $Q = A^\dagger$, we see that the iterations in Algorithm 7 and Algorithm 10 are the same. We can use this insight to show the equivalence of the EC algorithm and the (unweighted) spectral learning algorithm.

Proposition 56. *If X, Y and the model rank d are chosen such that the singular values σ_i of $\hat{F}_{Y,X}$ satisfy $\sigma_d > \sigma_{d+1}$, then the EC Algorithm 10 and the spectral Algorithm 6 will give characterizer and indicator pairs $(C_{\text{EC}}, Q_{\text{EC}})$ and $(C_{\text{SPEC}}, Q_{\text{SPEC}})$, respectively, that lead to equivalent model estimates via the learning Equations (2).*

Proof. The condition $\sigma_d > \sigma_{d+1}$ assures that $\text{rank}(\hat{F}_{Y,X}) \geq d$ and that its best rank- d approximation is given uniquely by its d -truncated SVD $U_d S_d V_d^\top$. Because the (unweighted) alternating projections Algorithm 7 is globally convergent [Tipping and Bishop, 1999], we must therefore have $BA = U_d S_d V_d^\top$ as well as $C_{\text{EC}} = B^\dagger$ and $Q_{\text{EC}} = A^\dagger$ upon convergence. This implies that $B = U_d \rho$ for some non-singular $\rho \in \mathbb{R}^{d \times d}$, and therefore $C_{\text{EC}} = (U_d \rho)^\dagger = \rho^{-1} U_d^\top = \rho^{-1} C_{\text{SPEC}}$, as well as $A = (U_d \rho)^\dagger U_d S_d V_d^\top = \rho^{-1} S_d V_d^\top$ and $Q_{\text{EC}} = (\rho^{-1} S_d V_d^\top)^\dagger = V_d S_d^{-1} \rho = Q_{\text{SPEC}} \rho$, and the claim follows by Lemma 42. $\blacksquare$

Moreover, the following Proposition 58 shows that a solution to the EC optimization problem from Equation (15) is readily obtained via the d -truncated SVD of $\hat{F}_{Y,X}$. This is a new result that leads to a non-iterative EC algorithm that is much simpler than the non-iterative *improved EC* algorithm previously developed by Zhao et al. [2009b]. Interestingly, this novel non-iterative EC algorithm turns out to be essentially the spectral learning Algorithm 6, which means we have discovered an alternative derivation of the spectral learning Algorithm 6. First, we require a technical lemma.

5. LEARNING SEQUENTIAL SYSTEMS

Lemma 57. For non-negative real numbers $d_1 \geq \dots \geq d_n \geq 0$ and $0 \leq s_1 \leq \dots \leq s_n$, define $D = \mathbf{D}(d_1, \dots, d_n)$ and $S = \mathbf{D}(s_1, \dots, s_n)$. Let $U \in \mathbb{R}^{n \times n}$ be an orthogonal matrix, i.e., $U^\top U = UU^\top = \mathbf{I}_n$. Then $\|DUS\|_F \geq \|DS\|_F$.

Proof. $\|DUS\|_F^2 = \sum_{i,j=1}^n (d_i u_{ij} s_j)^2$. Furthermore, $U^\top U = UU^\top = \mathbf{I}_n$ implies that $(*)$ $\sum_{i=1}^n u_{i,j}^2 = 1$ for all j and $\sum_{j=1}^n u_{i,j}^2 = 1$ for all i . We will show the slightly stronger claim that $\|DUS\|_F^2 \geq \|DS\|_F^2$ for any matrix U satisfying $(*)$. So from now on we assume that U merely satisfies $(*)$.

First note that if U is lower triangular, then $(*)$ implies that $U = \mathbf{I}_n$: $\sum_{i=1}^n u_{i,n}^2 = 1$ implies that $u_{n,n}^2 = 1$, since $u_{i,n}^2 = 0$ for $i < n$. Then $\sum_{j=1}^n u_{n,j}^2 = 1$ implies that $u_{n,j}^2 = 0$ for $j < n$, since $u_{n,n}^2 = 1$. That is, $U = \begin{bmatrix} U_{n-1} & 0 \\ 0 & 1 \end{bmatrix}$, and the condition $(*)$ must therefore hold for U_{n-1} as well. By iterating this argument, we get $U = \mathbf{I}_n$, in which case $\|DUS\|_F^2 = \|DS\|_F^2$.

So assume that U is not lower triangular. Consider a row-wise ordering of matrix positions via $\text{ord}(i, j) = (i-1)n + j$ and let $(i', j') = \arg \min_{(i,j)} \{\text{ord}(i, j) : j > i, u_{i,j} \neq 0\}$. Then i' is the first row of U to contain a non-zero element above the diagonal, and j' is the column index of the first such entry within the i' -th row. We call $\text{ord}(i', j')$ the *order* of U and note that the order of U is an integer less than n^2 .

Now consider the i' -th column of U . By the choice of i' we must have $\sum_{i=1}^{i'-1} u_{i,i'}^2 = 0$, and therefore $\sum_{i=i'+1}^n u_{i,i'}^2 = 1 - u_{i',i'}^2 = \sum_{j=1}^n u_{i',j}^2 - u_{i',i'}^2 \geq u_{i',j'}^2$. We can therefore find a vector v such that $v_i = 0$ for $i < i'$, $v_{i'} = -u_{i',j'}^2$, and $0 \leq v_i \leq u_{i,i'}^2$ for $i = i'+1, \dots, n$ such that $\sum_{i=i'+1}^n v_i = u_{i',j'}^2$. Let $U^2 = [u_{i,j}^2]_{i,j=1 \dots n}$ be the matrix of element-wise squares of entries in U , and let $\tilde{U}^2$ be obtained by subtracting the vector v from the i' -th column of U^2 and adding v to the j' -th column of U^2 . Let $\tilde{U}$ be the matrix of element-wise square roots of entries in $\tilde{U}^2$.

We can easily check that all entries in $\tilde{U}^2$ are non-negative, so that this is well-defined. Also $\tilde{U}$ satisfies $(*)$, since $\sum_{i=1}^n v_i = 0$ by construction and adding such a vector to one column of $\tilde{U}^2$ and subtracting from another does not change the row and column sums. Furthermore,

$$\|DUS\|_F^2 - \|D\tilde{U}S\|_F^2 = \sum_{i=1}^n (d_i^2 v_i s_{i'}^2 - d_i^2 v_i s_{j'}^2) = (s_{i'}^2 - s_{j'}^2) \left(d_{i'}^2 v_{i'} + \sum_{i=i'+1}^n d_i^2 v_i \right).$$

Now $s_{i'}^2 - s_{j'}^2 \leq 0$ since $j' > i'$, and $\sum_{i=i'+1}^n d_i^2 v_i \leq d_{i'}^2 \sum_{i=i'+1}^n v_i = d_{i'}^2 u_{i',j'}^2$, while $d_{i'}^2 v_{i'} = -d_{i'}^2 u_{i',j'}^2$, so $(d_{i'}^2 v_{i'} + \sum_{i=i'+1}^n d_i^2 v_i) \leq 0$. This shows that $\|DUS\|_F^2 \geq \|D\tilde{U}S\|_F^2$.

5.4 Selecting the Characterizer and Indicator

Finally, the order of $\tilde{U}$ is larger than the order of U , as we have eliminated the non-zero element of lowest order above the diagonal in U and in turn have introduced only non-zero elements above the diagonal in positions of higher order (in rows below the i' -th), or none at all.

By iterating this construction we arrive at a lower triangular matrix U^* that satisfies $(*)$, hence $U^* = \mathbf{I}_n$, and $\|DUS\|_F^2 \geq \|DU^*S\|_F^2 = \|DS\|_F^2$. $\blacksquare$

Proposition 58. *Let $U_d S_d V_d^\top$ be the d -truncated SVD of $\hat{F}_{Y,X}$. Then $C^* = S_d^{-\frac{1}{2}} U_d^\top$ and $Q^* = (C^* \hat{F}_{Y,X})^\dagger = V_d S_d^{-\frac{1}{2}}$ are a solution to the EC optimization problem in Equation (15) – provided a solution exists at all, i.e., $\text{rank}(\hat{F}_{Y,X}) \geq d$.*

Proof. Assume $r = \text{rank}(\hat{F}_{Y,X}) \geq d$ and let $USV^\top = \hat{F}_{Y,X}$ be the full SVD of $\hat{F}_{Y,X}$. We can verify that indeed $Q^* = (C^* \hat{F}_{Y,X})^\dagger = (S_d^{-\frac{1}{2}} U_d^\top U S V^\top)^\dagger = (S_d^{\frac{1}{2}} V_d^\top)^\dagger = V_d S_d^{-\frac{1}{2}}$, which implies that $C^* \hat{F}_{Y,X} Q^* = (S_d^{\frac{1}{2}} V_d^\top)(V_d S_d^{-\frac{1}{2}}) = \mathbf{I}_d$, as required. Furthermore, $\|C^*\|_F \|Q^*\|_F = \|S_d^{-\frac{1}{2}}\|_F^2 = \sum_{i=1}^d \sigma_i^{-1}$, where the σ_i are the singular values of $\hat{F}_{Y,X}$, which are also the diagonal elements of S . We will show that this is indeed the minimum of $\|C\|_F \|Q\|_F$ subject to $C \hat{F}_{Y,X} Q = \mathbf{I}_d$.

Using the substitution $C = C' U^\top$ and $Q = V Q'$, we can see that minimizing $\|C\|_F \|Q\|_F$ subject to $C \hat{F}_{Y,X} Q = \mathbf{I}_d$ is equivalent to minimizing $\|C'\|_F \|Q'\|_F$ subject to $C' S Q' = \mathbf{I}_d$ and that this will have the same minimal value. Let C'_r , Q'_r , and S_r be truncated versions of C' , Q' , and S that consist of the first r columns, rows or rows and columns, respectively. Then minimizing $\|C'_r\|_F \|Q'_r\|_F$ subject to $C'_r S_r Q'_r = \mathbf{I}_d$ is equivalent and has the same minimal value, because $C' S Q' = C'_r S_r Q'_r$ (since $\sigma_i = 0$ for $i > r$) and the additional columns in C' and rows in Q' are best set to zero.

Assume now that C'_r and Q'_r minimize $\|C'_r\|_F \|Q'_r\|_F$ subject to $C'_r S_r Q'_r = \mathbf{I}_d$. We can select $Q'_r = (C'_r S_r)^\dagger$, as this minimizes $\|C'_r\|_F \|Q'_r\|_F$ subject to $C'_r S_r Q'_r = \mathbf{I}_d$ for a given C'_r . It remains to show that $\|C'_r\|_F \|Q'_r\|_F \geq \|C^*\|_F \|Q^*\|_F = \sum_{i=1}^d \sigma_i^{-1}$.

Let $LDR^\top = C'_r S_r$ be the SVD of $C'_r S_r$. Then $C'_r = LDR^\top S_r^{-1}$, and $Q'_r = (C'_r S_r)^\dagger = RD^\dagger L^\top$. Let $d_1, \dots, d_d$ be the diagonal elements of D and let D_r be the $r \times r$ matrix obtained by extending D with zero rows. Then

$$\begin{aligned} \|C'_r\|_F^2 &= \|LDR^\top S_r^{-1}\|_F^2 = \|D_r R^\top S_r^{-1}\|_F^2 \stackrel{\text{Lem. 57}}{\geq} \|D_r S_r^{-1}\|_F^2 = \sum_{i=1}^d d_i^2 \sigma_i^{-2}, \\ \|Q'_r\|_F^2 &= \|RD^\dagger L^\top\|_F^2 = \|D^\dagger\|_F^2 = \sum_{i=1}^d d_i^{-2}. \end{aligned}$$

5. LEARNING SEQUENTIAL SYSTEMS

Multiplying these expressions and substituting $d_i^2 = a_i^2 \sigma_i$, we obtain

$$\begin{aligned}
\|C'_r\|_F^2 \|Q'_r\|_F^2 &= \left(\sum_{i=1}^d a_i^2 \sigma_i^{-1} \right) \left(\sum_{i=1}^d a_i^{-2} \sigma_i^{-1} \right) \\
&= \sum_{i=1}^d \sigma_i^{-2} + \sum_{\substack{i,j=1 \\ i < j}}^d \left(\frac{a_i^2}{a_j^2} + \frac{a_j^2}{a_i^2} \right) \sigma_i^{-1} \sigma_j^{-1} \\
&= \sum_{i=1}^d \sigma_i^{-2} + \sum_{\substack{i,j=1 \\ i < j}}^d \left(\left(\frac{a_i}{a_j} - \frac{a_j}{a_i} \right)^2 + 2 \right) \sigma_i^{-1} \sigma_j^{-1} \\
&\geq \left(\sum_{i=1}^d \sigma_i^{-1} \right)^2, \quad \text{since } \sum_{\substack{i,j=1 \\ i < j}}^d \left(\frac{a_i}{a_j} - \frac{a_j}{a_i} \right)^2 \sigma_i^{-1} \sigma_j^{-1} \geq 0.
\end{aligned}$$

So we can conclude that $\|C'_r\|_F \|Q'_r\|_F \geq \sum_{i=1}^d \sigma_i^{-1}$. Therefore, C^* and Q^* are in fact a minimal solution to the optimization problem (15). $\blacksquare$

Comparing the resulting selection of characterizer $C^* = S_d^{-\frac{1}{2}} U_d^\top$ and indicator $Q^* = V_d S_d^{-\frac{1}{2}}$, where $U_d S_d V_d^\top$ is the d -truncated SVD of $\hat{F}_{Y,X}$, to the selection $C_{\text{SPEC}} = U_d^\top$ and $Q_{\text{SPEC}} = V_d S_d^{-1}$ of the spectral Algorithm 6, we see that these rely (implicitly in the case of EC) on slightly different basis representations of the principal subspace of $\hat{F}_{Y,X}$, namely $U_d S_d^{\frac{1}{2}}$ in the case of the EC algorithm and U_d in the case of the spectral Algorithm 6, which will result in equivalent model estimates by Lemma 42.

Note, however, that it is also possible to derive different bounds on the relative approximation error than given in Proposition 55, which can lead to a selection of C and Q by the EC principle that is not equivalent to the spectral learning Algorithm 6. An example is the *improved EC* algorithm, which uses a slightly different bound. While this gives non-equivalent model estimates, they do appear to be of comparable quality [Zhao et al., 2009b].

5.4.6 The Efficiency Sharpening Algorithm

The efficiency sharpening (ES) algorithm has previously been worked out only for the case of stationary stochastic processes and “traditional” OOMs where $\sigma = \mathbb{1}^\top$. Here we give a streamlined account of the ES principle that is more general than in the original work, and we establish a connection to the row and column weighted spectral Algorithm 9. The (generalized) ES Algorithm 11 we present here is also applicable to

5.4 Selecting the Characterizer and Indicator

learning SMA, IO-OOMs, and PSRs from data.

The idea of the ES learning algorithm by Jaeger et al. [2006b] is to view the learning Equations (2) as a model estimator parameterized by C (and Q), and to select C such that the mean squared error of the resulting estimator is minimized while still being a consistent estimator. Furthermore, this optimal choice of C is derived from knowledge of a model $\mathcal{M}$ for f , or in practice from a previous estimate thereof. To make this approach tractable, some simplifying assumptions are made.

First, a simplified version of the learning Equations (2) is considered, where the indicator is taken to be $Q = (CF_{Y,X})^\dagger$. This leads to operator estimates

$$\hat{\tau}_z = C\hat{F}_{zY,X}(C\hat{F}_{Y,X})^\dagger.$$

Jaeger et al. [2006b] now argue that due to the (pseudo-)inversion, the mean squared error of $\hat{\tau}_z$ is dominated by the mean squared error of the factor $C\hat{F}_{Y,X}$. The ES algorithm therefore strives to find an admissible C such that the mean squared error of $C\hat{F}_{Y,X}$ is minimized – assuming knowledge of a model $\mathcal{M}$ for f . This is solved by the following proposition, which we state here in a more general form than in the original work [Jaeger et al., 2006b], and with new and more rigorous proof.

Proposition 59. *Let $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_\varepsilon)$ be a minimal d -dimensional SS for a function $f : \Sigma^* \rightarrow \mathbb{R}$. Let X and Y be sets of indicative and characteristic words such that $\text{rank}(F_{Y,X}) = d$ and let $\hat{f}(\bar{x})$ for $\bar{x} \in \Sigma^*$ be unbiased and uncorrelated estimators for $f(\bar{x})$ with $v_{\bar{y}} = \sum_{\bar{x} \in X} \mathbb{V}[\hat{f}(\bar{x}\bar{y})] > 0$ for all $\bar{y} \in Y$. Let $\Pi = [(\sigma\tau_{\bar{y}})^\top]_{\bar{y} \in Y}^\top$, $\Phi = [\tau_{\bar{x}}\omega_\varepsilon]_{\bar{x} \in X}$ and*

$$C^* = \Pi^\top \mathbf{D}(v_Y^\top)^{-1}, \quad \text{where } v_Y^\top = \left[\sum_{\bar{x} \in X} \mathbb{V}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{y} \in Y}^\top.$$

Then the mean squared error $\text{MSE}[C\hat{F}_{Y,X}]$ is minimized by the characterizer C^* among all characterizers C of the form $C^* + G$ that satisfy $G\Pi = 0$.

Furthermore, C^* is an admissible characterizer, i.e., $C^*F_{Y,X}$ has full rank d , and every admissible characterizer C can be written as $\rho(C^* + G)$ for some non-singular $\rho \in \mathbb{R}^{d \times d}$ and G such that $G\Pi = 0$.

Proof. First, for any characterizer C we can calculate:

$$\begin{aligned} \text{MSE}[C\hat{F}_{Y,X}] &= \mathbb{E} \left[\|C\hat{F}_{Y,X} - CF_{Y,X}\|_F^2 \right] \stackrel{*}{=} \mathbb{E} \left[\|C\hat{F}_{Y,X} - \mathbb{E}[C\hat{F}_{Y,X}]\|_F^2 \right] \\ &= \sum_{\bar{x} \in X} \mathbb{E} \left[\|C[\hat{F}_{Y,X}]_{\bar{x}} - \mathbb{E}[C[\hat{F}_{Y,X}]_{\bar{x}}]\|_F^2 \right] = \sum_{\bar{x} \in X} \text{tr} \left(\mathbb{V}[C[\hat{F}_{Y,X}]_{\bar{x}}] \right) \end{aligned}$$

5. LEARNING SEQUENTIAL SYSTEMS

$$\begin{aligned}
& \stackrel{\text{Lem 51}}{=} \sum_{\bar{x} \in X} \text{tr} \left(C \mathbb{V}[\hat{F}_{Y,X}]_{\bar{x}} C^\top \right) & \stackrel{**}{=} \sum_{\bar{x} \in X} \text{tr} \left(C \mathbf{D} \left(\left[\mathbb{V}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{y} \in Y}^\top \right) C^\top \right) \\
& = \text{tr} \left(C \mathbf{D}(v_Y^\top) C^\top \right) & = \|C \mathbf{D}(v_Y^\top)^{\frac{1}{2}}\|_{\text{F}}^2
\end{aligned}$$

where we have used unbiasedness for (*) and uncorrelatedness for (**). For a characterizer of the form $C^* + G = \Pi^\top \mathbf{D}(v_Y^\top)^{-1} + G$ with $G\Pi = 0$ we therefore obtain

$$\begin{aligned}
\text{MSE}[(C^* + G)\hat{F}_{Y,X}] &= \|(C^* + G) \mathbf{D}(v_Y^\top)^{\frac{1}{2}}\|_{\text{F}}^2 = \text{tr} \left((C^* + G) \mathbf{D}(v_Y^\top) (C^* + G)^\top \right) \\
&= \|C^* \mathbf{D}(v_Y^\top)^{\frac{1}{2}}\|_{\text{F}}^2 + \|G \mathbf{D}(v_Y^\top)^{\frac{1}{2}}\|_{\text{F}}^2 + 2 \underbrace{\text{tr}(G \mathbf{D}(v_Y^\top) C^{*\top})}_{=G\Pi=0} \\
&\geq \|C^* \mathbf{D}(v_Y^\top)^{\frac{1}{2}}\|_{\text{F}}^2.
\end{aligned}$$

Now $d = \text{rank}(F_{Y,X}) = \text{rank}(\Pi\Phi) = \text{rank}(\Pi) = \text{rank}(\Phi)$, and $\mathbf{D}(v_Y^\top)^{-\frac{1}{2}}$ is invertible, so $C^*\Pi = (\mathbf{D}(v_Y^\top)^{-\frac{1}{2}}\Pi)^\top (\mathbf{D}(v_Y^\top)^{-\frac{1}{2}}\Pi) \in \mathbb{R}^{d \times d}$ is invertible, and C^* therefore admissible.

Finally, let C be any admissible characterizer. Then $C\Pi \in \mathbb{R}^{d \times d}$ must be invertible. Setting $\rho = (C\Pi)(C^*\Pi)^{-1}$ and $G = \rho^{-1}(C - \rho C^*)$ we can easily verify that $G\Pi = 0$ and $C = \rho(C^* + G)$. ■

Note that the characterizers $C^* + G$ and $\rho(C^* + G)$ will lead to equivalent models by Lemma 42. Therefore, if the characterizer C^* is best among the class of characterizers $C^* + G$ where $G\Pi = 0$ then it is also the overall best choice.

The above Proposition 59 requires a suitable pre-selection of indicative and characteristic words X and Y , as well as knowledge of the correct model dimension d , such that $F_{Y,X}$ has rank d . In practice, this cannot be guaranteed, but the methods presented in the previous Sections 5.2 and 5.3 can be used instead. Also, the estimates $\hat{f}(\bar{x})$ may not be unbiased and uncorrelated, meaning that the characterizer C^* will only approximate the theoretically optimal characterizer. A further requirement is that all entries in $v_Y^\top$, which measure the variance sum of each row of $\hat{F}_{Y,X}$, must be strictly positive, since their inverse is used to weight the corresponding rows of $\hat{F}_{Y,X}$ (and $\hat{F}_{zY,X}$ and $\hat{F}_Y^\top$). If a row has zero variance, then it makes sense to use a very large weight. One can achieve this by enforcing some minimum value for the variance $\mathbb{V}[\hat{f}(\bar{x})]$ of any estimate $\hat{f}(\bar{x})$, for instance as done in Section 5.1.1. Furthermore, to compute the characterizer C^* from Proposition 59, we need to be able to compute the variances $\mathbb{V}[\hat{f}(\bar{x})]$ of the estimates $\hat{f}(\bar{x})$ occurring in $\hat{F}_{Y,X}$. We can approximate these variances by the methods presented in Section 5.1.1, but may here use the value $f(\bar{x})$ computed from the known model $\mathcal{M}$ in place of the estimates $\hat{f}(\bar{x})$.

5.4 Selecting the Characterizer and Indicator

Now, the ES algorithm does not actually use the indicator $Q = (C^* \hat{F}_{Y,X})^\dagger$, but instead the indicator

$$Q^* = \mathbf{D}(v_X)^{-\frac{1}{2}} \left(C^* \hat{F}_{Y,X} \mathbf{D}(v_X)^{-\frac{1}{2}} \right)^\dagger, \quad \text{where } v_X = \left[\sum_{y \in Y} \mathbb{V}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{x} \in X}.$$

It is easy to see that this amounts to performing column-weighted linear regression to compute the operators $\hat{\tau}_z$ from the argument-value pairs $C^* \hat{F}_{Y,X}$ and $C^* \hat{F}_{zY,X}$ via the learning Equations (2), where the inverse variance sum of each column of $\hat{F}_{Y,X}$ is used as column weight, which needs to be approximated, analogous to $v_Y^\top$, either from the known model $\mathcal{M}$, or as described in Section 5.1.1 from the available data. Again, we must ensure that all entries in v_X are strictly positive.

In practice a true model $\mathcal{M}$ is unknown. Instead, the ES Algorithm 11 is iterated starting from some initial model estimate that may either be purely random or computed by some other learning algorithm, for instance a spectral algorithm. The rationale of iterating the ES Algorithm 11 is that each iteration should compute a new estimate with lower mean squared error and therefore higher statistical efficiency, because the new estimate is based on a characterizer and indicator derived from an increasingly accurate previous model estimate, hence the name *efficiency sharpening*. However, this iteration comes with no convergence guarantees and may not converge at all. Therefore, a fixed number of iterations are computed, and the model estimate with the highest likelihood measured on the training data is selected. This procedure has been found to achieve good results in practice using simple initial model estimates and very few iterations (at most 5 typically suffice) with results that are of similar quality to those obtained by spectral algorithms [Jaeger et al., 2006b; Zhao et al., 2009a,b].

Algorithm 11: The (generalized) efficiency sharpening (ES) algorithm

In : A d -dimensional model estimate $\hat{\mathcal{M}} = (\hat{\sigma}, \{\hat{\tau}_z\}, \hat{\omega}_\varepsilon)$, estimates $\hat{F}_{Y,X}$, and variance estimates $\hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})]$ (e.g., $\hat{\mathbb{V}}_{2,0}[\hat{f}(\bar{x}\bar{y})]$) for the entries in $\hat{F}_{Y,X}^1$

Out: $\hat{C}^*$, $\hat{Q}^*$ for use in the learning Equations (2)

$$\begin{aligned} \hat{v}_Y^\top &\leftarrow \left[\sum_{\bar{x} \in X} \hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{y} \in Y}^\top, & \hat{v}_X &\leftarrow \left[\sum_{y \in Y} \hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{x} \in X} \\ \hat{\Pi} &\leftarrow [(\hat{\sigma} \hat{\tau}_{\bar{y}})^\top]_{\bar{y} \in Y}^\top \\ \hat{C}^* &\leftarrow \hat{\Pi}^\top \mathbf{D}(\hat{v}_Y^\top)^{-1}, & \hat{Q}^* &\leftarrow \mathbf{D}(\hat{v}_X)^{-\frac{1}{2}} \left(\hat{C}^* \hat{F}_{Y,X} \mathbf{D}(\hat{v}_X)^{-\frac{1}{2}} \right)^\dagger \end{aligned}$$

Note that the runtime complexity of Algorithm 11 is $\mathcal{O}(d|Y|(l_Y d + |X|))$, where

5. LEARNING SEQUENTIAL SYSTEMS

l_Y is the average length of words in Y , since the computation of $\hat{\Pi}$ requires time in $\mathcal{O}(l_Y|Y|d^2)$, while the computation of Q^* from C^* takes $\mathcal{O}(d|X||Y|)$ time. As l_Y is typically small, $d < |X|$, and only few iterations are required, this is on par with the runtime complexity of the (row and column weighted) spectral learning algorithm.

The ES Algorithm 11 is, in fact, a variant of the row and column weighted spectral Algorithm 9 presented in Section 5.4.4. The difference lies in the way a basis $\hat{B}$ for the column space of $F_{Y,X}$ is estimated. While the spectral algorithms in Step (i) obtain a factorization $F_{Y,X} \approx \hat{B}\hat{A}$ from the matrix estimate $\hat{F}_{Y,X}$ by computing its best (weighted) rank- d approximation $\hat{B}\hat{A}$, the ES algorithm instead uses a previous model estimate $\hat{\mathcal{M}}$ to compute $\hat{B} = \hat{\Pi}$ and $\hat{\Phi}$, since $F_{Y,X} \approx \hat{\Pi}\hat{\Phi}$.

To see that this is the only difference, let us use $\hat{B} = \hat{\Pi}$ as a basis estimate for the column space of $F_{Y,X}$, and then follow Steps (ii) and (iii) of the row and column weighted spectral algorithm from Section 5.4.4. According to Equation (14) this yields the characterizer and indicator

$$\begin{aligned} C &= (\hat{B}^\top \mathbf{D}(w_Y^\top) \hat{B})^{-1} \hat{B}^\top \mathbf{D}(w_Y^\top) \\ &= \rho^{-1} \hat{\Pi}^\top \mathbf{D}(v_Y^\top)^{-1} = \rho^{-1} \hat{C}^*, \quad \text{and} \\ Q &= \mathbf{D}(w_X)^{\frac{1}{2}} (C \hat{F}_{Y,X} \mathbf{D}(w_X)^{\frac{1}{2}})^\dagger \\ &= \mathbf{D}(v_X)^{-\frac{1}{2}} (\hat{C}^* \hat{F}_{Y,X} \mathbf{D}(v_X)^{-\frac{1}{2}})^\dagger \rho = \hat{Q}^* \rho, \end{aligned}$$

where $\rho = \hat{B}^\top \mathbf{D}(v_Y^\top)^{-1} \hat{B}$ and we have used the fact that the ES Algorithm 11 and the row and column weighted spectral Algorithm 9 employ essentially the same row and column weights¹, namely $\mathbf{D}(w_Y^\top) = |X| \mathbf{D}(v_Y^\top)^{-1}$ and $\mathbf{D}(w_X) = |Y| \mathbf{D}(v_X)^{-1}$. This recovers the characterizer $\hat{C}^*$ and indicator $\hat{Q}^*$ employed by the ES Algorithm 11 up to the non-singular transformation ρ , which leads to an equivalent model estimate by Lemma 42.

Finally, we will further examine a particular variance approximation $\hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})]$ that can be used to compute $\hat{v}_Y^\top$ and leads to a special form of the characterizer C^* but that is valid only for the estimators $\hat{f}_\pi$ from Remark 37 for the case of stationary stochastic processes. If we modify the variance estimate from Remark 39 to use the known value $f(\bar{x}\bar{y})$ in place of the Bayesian estimate $\tilde{p}(\bar{x}\bar{y})$, we get $\hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})] = f(\bar{x}\bar{y})(1 - f(\bar{x}\bar{y}))/n$, which we may further approximate by $f(\bar{x}\bar{y})/n$, as in practice the values of $f(\bar{x}\bar{y})$ will typically be small for most sequences $\bar{x}\bar{y}$. Also, the division by n is superfluous, as it cancels via the learning Equations (2), so we may use the variance approximation $\hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})] = f(\bar{x}\bar{y})$. In the case of a stationary stochastic process and a choice of

¹Alternatively, the ES algorithm can use a previous model estimate $\hat{\mathcal{M}}$ to approximate $\mathbb{V}[\hat{f}(\bar{x}\bar{y})]$.

indicative words X that partition Σ^l (or $\Sigma^{\leq l}$) for some l we have $\sum_{\bar{x} \in X} f(\bar{x}\bar{y}) = f(\bar{y})$ (or $= (l+1)f(\bar{y})$, but again $(l+1)$ will cancel in the learning Equations (2)), and we then obtain the simple variance approximation

$$\hat{v}_Y^\top = \left[\sum_{\bar{x} \in X} \hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})] \right]_{\bar{y} \in Y}^\top = [f(\bar{y})]_{\bar{y} \in Y}^\top = \Pi \omega_\varepsilon.$$

In this case the columns of the characterizer C^* have the form $[C^*]_{\bar{y}} = \sigma \tau_{\bar{y}} / \sigma \tau_{\bar{y}} \omega_\varepsilon$, which are just normalized co-states of the model $\mathcal{M}$. This is essentially the original version given by Jaeger et al. [2006b], who viewed these co-states as states of a time-reversed model instead and, therefore, named the resulting characterizer the *reverse characterizer*. This special form of the reverse characterizer is further exploited by Jaeger et al. [2006b] to compute an approximation of $C^* \hat{F}_{\Sigma^*, X}$ and $C^* \hat{F}_{z\Sigma^*, X}$ directly from a suffix-tree representation of the training data, leading to an efficient *suffix-tree ES algorithm* that implicitly uses as characteristic words X all words from Σ^* that occur in the training data. For completeness, we should mention that Jaeger et al. [2006b] use yet another approximation for $\hat{v}_X$ that is computed from the training data, namely $\hat{\mathbb{V}}[\hat{f}(\bar{x}\bar{y})] = \hat{f}(\bar{x}\bar{y})$, leading to $\hat{v}_X = [\sum_{\bar{y} \in Y} \hat{f}(\bar{x}\bar{y})]_{\bar{x} \in X}$.

5.5 Computational Complexity

We have already noted the computational complexity for most of the steps involved in the general learning Algorithm 4. Here, we put these results together to obtain a complete picture of the runtime requirement for the general learning Algorithm 4.

Assume we are given training data consisting of N symbols over an alphabet Σ . First, we will compute a radix tree or generalized suffix tree representation of the training data and annotate the nodes by their leaf counts using a depth-first tree traversal, which requires $\mathcal{O}(N \log |\Sigma|)$ time and $\mathcal{O}(N)$ space if we use balanced search trees to represent the parent-child relationships in the tree. The reason to use balanced search trees is that this allows us to look up a word $\bar{x}$ in the tree in time $\mathcal{O}(|\bar{x}| \log |\Sigma|)$, which means that computing the estimate $\hat{f}(\bar{x})$ as well as the corresponding variance estimate by the methods in Section 5.1 takes time in $\mathcal{O}(|\bar{x}| \log |\Sigma|)$ as well¹.

Given this tree representation of the training data, we then select sets X and Y of indicative and characteristic words using Algorithm 5 in time $\mathcal{O}(N \log N + |X| + |Y|)$,

¹Alternatively, we may speed up the tree generation to $\mathcal{O}(N)$ if we are willing to spend $\mathcal{O}(|\bar{x}| |\Sigma|)$ time to compute the estimates for each word $\bar{x}$

5. LEARNING SEQUENTIAL SYSTEMS

or alternatively $\mathcal{O}((|X| + |Y|)|\Sigma| \log(|X| + |Y|)|\Sigma|)$ if we use a setting of $L_{\min} = 0$.

Computing the matrices $\hat{F}_{Y,X}$, $\hat{F}_{zY,X}$, $\hat{F}_Y^\top$, and $\hat{F}_X$, as well as the corresponding variance estimates, then requires $\mathcal{O}(|\Sigma||X||Y|(l_X + l_Y) \log |\Sigma|)$ time, where l_X and l_Y are the average length of words in X and Y , respectively.

Estimating the target dimension d in Step 3 by computing the numerical rank of the matrix $\hat{F}_{Y,X}$ or its scaled version $\tilde{F}_{Y,X}$ requires $\mathcal{O}(d|X||Y|)$ operations as explained at the end of Section 5.3.

Computing suitable characterizer and indicator matrices C and Q for Step 4 takes time in $\mathcal{O}(d|X||Y|)$ in the case of (row and column weighted) spectral learning, and $\mathcal{O}(d|Y|(l_Y d + |X|))$ in the case of ES, while solving the learning Equations (2) in Step 5 takes $\mathcal{O}(|\Sigma|d|X||Y|)$ time. Using element-wise weighted learning to obtain a model estimate through Steps 4 and 5 takes time in $\mathcal{O}((|\Sigma| + K)d^2|X||Y|)$, where K is the number of iterations required for the alternating projections algorithm to converge, while weighted spectral learning incurs a runtime complexity of $\mathcal{O}((|\Sigma| + K)d^2|X||Y| + |\Sigma|d^4(d^2 + |X|))$.

Let us make the simplifying assumptions that $|X| = |Y|$, that l_X, l_Y and $\log |\Sigma|$ are small enough that we can neglect them as constants and that we select indicative and characteristic words using a parameter setting of $L_{\min} = 0$. We then obtain the following Table 3 of runtimes for the main steps of the general learning Algorithm 4.

Task	Runtime complexity
tree representation of data	$\mathcal{O}(N)$
selection of words X and Y	$\mathcal{O}(\Sigma X \log X)$
computation of all estimates	$\mathcal{O}(\Sigma X ^2)$
estimation of target dimension	$\mathcal{O}(d X ^2)$
(row/column weighted) spectral learning, ES	$\mathcal{O}(\Sigma d X ^2)$
element-wise weighted spectral learning	$\mathcal{O}((\Sigma + K)d^2 X ^2)$
weighted spectral learning	$\mathcal{O} \left(\begin{array}{l} (\Sigma + K)d^2 X ^2 \\ + \Sigma d^2d^2(d^2 + X) \end{array} \right)$

Table 3: Runtime complexity of the main steps of the general learning Algorithm 4 under mild simplifying assumptions.

Note that in terms of these complexity results, the element-wise weighted learning is slower than (row and column weighted) spectral learning and ES by a factor of d , while weighted spectral learning is only slower than element-wise spectral learning if $d^2 > |X|$. If, however, $d \approx |X|$, then weighted spectral learning is slower than element-wise weighted spectral learning by a factor of d^2 . The additional complexity of the element-wise weighted and even more so of the weighted spectral learning algorithm,

therefore, primarily depends on the size of the target dimension d . For large target dimensions d , weighted spectral learning may be an impractical algorithm. In fact, weighted spectral learning requires computing the Cholesky decomposition of a matrix of size $d^2 \times d^2$, which has a memory requirement of about $8d^4$ bytes when storing the entries as 64-bit floating point numbers. In the case of $d = 192$ the memory requirement for this Cholesky decomposition alone is about 10 GB, which we will regard as the practical limit on current standard hardware, while a setting of $d = 256$ already requires about 32 GB memory.

5.6 Normalization, Stabilization, Sampling

Above we have discussed in detail the steps required to obtain a model estimate $\hat{\mathcal{M}}$ from sample data via the general learning Algorithm 4, which allows us to compute $f_{\hat{\mathcal{M}}}(\bar{x})$ for $\bar{x} \in \Sigma^*$ via Equation (1). One remaining issue that we have mentioned but not discussed is that the estimated model $\hat{\mathcal{M}}$ may not satisfy the conditions of a (controlled) stochastic process or stochastic language, especially $f_{\hat{\mathcal{M}}} \geq 0$, which is a necessary but undecidable requirement. To deal with this, we adapt a normalization procedure developed by Jaeger et al. [2006b, Appendix J] to also cover the case of models for controlled stochastic processes. At the same time, this allows us to sample efficiently from the process described by a SS. We later explain how this can be used for models of stochastic languages as well.

Let us first consider only the case of (controlled) stochastic processes. To ensure that $f_{\hat{\mathcal{M}}} \geq 0$, we will alter the way that the function values $f_{\hat{\mathcal{M}}}(\bar{x}) = \sigma\tau_{\bar{x}}\omega_{\varepsilon}$ are computed. Recall that for a (controlled) stochastic process f , we have

$$f(x_1 \dots x_n) = \prod_{k=1}^n f(x_k | \bar{x}_{k-1})$$

$$\text{where } \bar{x}_k = x_1 \dots x_k \text{ and } f(\bar{y} | \bar{x}) = \begin{cases} \frac{f(\bar{x}\bar{y})}{f(\bar{x})} & \text{if } f(\bar{x}) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

If f is described by a SS $\mathcal{M} = (\sigma, \{\tau_z\}, \omega_{\varepsilon})$, we can compute $f(\bar{y} | \bar{x}) = \sigma\tau_{\bar{y}}\tau_{\bar{x}}\omega_{\varepsilon} / \sigma\tau_{\bar{x}}\omega_{\varepsilon}$ from the normalized state $\dot{\omega}_{\bar{x}} = \tau_{\bar{x}}\omega_{\varepsilon} / \sigma\tau_{\bar{x}}\omega_{\varepsilon}$, so we can rewrite the above Equation (16) via the iteration

$$f(x_1 \dots x_k) = f(x_k | \dot{\omega}_{\bar{x}_{k-1}}) f(x_1 \dots x_{k-1}) \quad \text{with } f(z | \dot{\omega}) = \sigma\tau_z\dot{\omega},$$

$$\dot{\omega}_{\bar{x}_k} = \tau_{x_k}\dot{\omega}_{\bar{x}_{k-1}} / \sigma\tau_{x_k}\dot{\omega}_{\bar{x}_{k-1}} \quad f(\varepsilon) = 1, \dot{\omega}_{\varepsilon} = \omega_{\varepsilon}.$$

5. LEARNING SEQUENTIAL SYSTEMS

At each time step we can normalize the values $f(z|\dot{\omega})$ for every $z \in \Sigma$ such that these lie in the interval $[p_{\min}, 1]$ for some parameter $p_{\min} \geq 0$ and satisfy property (ii) of a (controlled) stochastic process, namely $\forall a \in \Sigma_I : \sum_{o \in \Sigma_O} f(ao|\dot{\omega}) = 1$. Specifying $p_{\min} > 0$ can be very useful to ensure that $f_{\mathcal{M}}(\bar{x}) > 0$ for all $\bar{x} \in \Sigma^*$. Additionally, we will perform a *state setback* if at some time step the function values $f_{\mathcal{M}}(z|\dot{\omega})$ need to be altered by more than a specified maximum error tolerance $e_{\max}$, or arithmetic underflow occurs during the state update, i.e., $\sigma\tau_z\dot{\omega} \leq \varepsilon_0$ for some zero margin ε_0 . A state setback is a stabilization heuristic that resets the current state $\dot{\omega}$ to the initial (stationary) state ω_ε and then replays a certain length l of the most recent history. The parameter $l_{\max}$ sets the maximum number of steps to attempt to replay during a state setback operation. This stabilization heuristic may be less effective and should therefore be omitted if ω_ε is not a stationary state, for instance in combination with certain input policies π , or when used for stochastic languages as explained later.

This is performed by Algorithm 12 and yields for any SS $\mathcal{M}$ a *normalized external function* $\dot{f}_{\mathcal{M}}$ that is guaranteed to be a (controlled) stochastic process:

Definition 60. *Let $\mathcal{M}$ be a SS. The normalized external function $\dot{f}_{\mathcal{M}}$ of the model $\mathcal{M}$ is defined by the equation*

$$\dot{f}_{\mathcal{M}}(x_1 \dots x_k) = \dot{f}_{\mathcal{M}}(x_k | \dot{\omega}_{\bar{x}_{k-1}}) \dot{f}_{\mathcal{M}}(x_1 \dots x_{k-1})$$

where the normalized and stabilized states $\dot{\omega}_{\bar{x}_k}$ as well as the normalized one step predictions $\dot{f}_{\mathcal{M}}(x_k | \dot{\omega}_{\bar{x}_{k-1}})$ are computed iteratively via Algorithm 12, starting from the initialization $\dot{\omega}_\varepsilon = \omega_\varepsilon$ and $\dot{f}_{\mathcal{M}}(\varepsilon) = 1$.

We have split Algorithm 12 into two steps, where Step 1 uses the current normalized state $\dot{\omega}$ and a given input symbol a to compute a normalized one-step prediction vector $\dot{p} = [\dot{f}_{\mathcal{M}}(ao|\dot{\omega})]_{o \in \Sigma_O}$, and Step 2 performs the normalized state update according to the current input-output symbol ao . This allows Algorithm 12 to be used to sample from $\dot{f}_{\mathcal{M}}$ as well: To obtain an initial realization of the (controlled) stochastic process described by $\dot{f}_{\mathcal{M}}$, at each time step t first determine the current input symbol a_t from the input policy π (which needs to be specified for sampling, since a controlled process only models the system output), then draw the current output symbol o_t according to the probability vector $\dot{p}$ computed by Step 1, and finally perform the state update via Step 2 according to $a_t o_t$.

It is easy to verify that for any SS $\mathcal{M}$ the normalized function $\dot{f}_{\mathcal{M}}$ will indeed describe a (controlled) stochastic process. Conversely, if $f_{\mathcal{M}}$ is a (controlled) stochastic

Algorithm 12: Normalization and stabilization (for (controlled) processes)

In : SS model $\mathcal{M}$, current normalized state $\dot{\omega}$, current input a (for controlled process), and parameters $p_{\min}$, $e_{\max}$, $l_{\max}$ and ε_0 .
 For Step 2: Current output symbol o .

Out : Step 1: Prediction vector $\dot{p} = [\dot{f}_{\mathcal{M}}(ao | \dot{\omega})]_{o \in \Sigma_O}$
 Step 2: Updated normalized state $\dot{\omega}$

Global: most recent history $\bar{z}$ for setback (initialized to ε)

```

1  $\dot{p}, e_{\dot{p}} \leftarrow \text{compute\_normalized\_p}(\mathcal{M}, \dot{\omega}, a)$ 
  while  $e_{\dot{p}} > e_{\max}$  and  $\dot{\omega} \neq \omega_{\varepsilon}$  do
     $\dot{\omega} \leftarrow \text{setback\_state}(\mathcal{M})$ 
     $\dot{p}, e_{\dot{p}} \leftarrow \text{compute\_normalized\_p}(\mathcal{M}, \dot{\omega}, a)$ 
2  $\dot{\omega} \leftarrow \tau_{ao}\dot{\omega}$ ;  $\bar{z} \leftarrow \bar{z}ao$ 
   if  $|\bar{z}| > l_{\max}$  then  $\bar{z} \leftarrow z_2 \dots z_{|\bar{z}|}$ 
   if  $\sigma\dot{\omega} \leq \varepsilon_0$  then  $\dot{\omega} \leftarrow \text{setback\_state}(\mathcal{M})$ 
   else  $\dot{\omega} \leftarrow \dot{\omega}/\sigma\dot{\omega}$ 

Procedure compute_normalized_p( $\mathcal{M}, \dot{\omega}, a$ )
   $p \leftarrow [\sigma\tau_{ao}\dot{\omega}]_{o \in \Sigma_O}$ 
  foreach  $o \in \Sigma_O$  do  $[p]_o \leftarrow \max(0, [p]_o - p_{\min})$ 
   $\dot{p}_{\Sigma_O} \leftarrow \sum_{o \in \Sigma_O} [\dot{p}]_o$ 
  if  $\dot{p}_{\Sigma_O} < \varepsilon_0$  then
    foreach  $o \in \Sigma_O$  do  $[\dot{p}]_o \leftarrow 1/|\Sigma_O|$ 
  else
    foreach  $o \in \Sigma_O$  do  $[\dot{p}]_o \leftarrow (1 - |\Sigma_O|p_{\min})[\dot{p}]_o/\dot{p}_{\Sigma_O} + p_{\min}$ 
   $e_{\dot{p}} \leftarrow \sqrt{\|p - \dot{p}\|^2/|\Sigma_O|}$ 
  return  $\dot{p}, e_{\dot{p}}$ 

```

Procedure setback_state($\mathcal{M}$)

```

   $\dot{\omega} \leftarrow \omega_{\varepsilon}$ 
  foreach  $z$  in  $\bar{z}$  do
     $\dot{\omega} \leftarrow \tau_z\dot{\omega}$ 
    if  $\sigma\dot{\omega} \leq \varepsilon_0$  then
       $\bar{z} \leftarrow z_2 \dots z_{|\bar{z}|}$ 
      return setback_state( $\mathcal{M}$ )
     $\dot{\omega} \leftarrow \dot{\omega}/\sigma\dot{\omega}$ 
   $\bar{z} \leftarrow z_2 \dots z_{|\bar{z}|}$ 
  return  $\dot{\omega}$ 

```

5. LEARNING SEQUENTIAL SYSTEMS

process, and we set $p_{\min} = 0$ and $\varepsilon_0 = 0$, then no normalization or stabilization will occur and $\dot{f}_{\mathcal{M}} = f_{\mathcal{M}}$.

Finally, this procedure can also be used to normalize and sample from an estimated SMA $\hat{\mathcal{M}}$ for a stochastic language using the equivalence between stochastic languages and stochastic processes with a termination symbol that we have established in Section 4.2.2. To convert $\hat{\mathcal{M}}$ into an estimate of a terminating OOM $\hat{\mathcal{M}}'$ as in Proposition 27, we may need to minimize $\hat{\mathcal{M}}$ using Algorithm 2 and then require that $\rho(\hat{\tau}_{\Sigma}) < 1$ for the spectral radius of $\hat{\tau}_{\Sigma} = \sum_{z \in \Sigma} \hat{\tau}_z$, so that we can compute $\hat{\sigma}(\mathbf{I}_d - \hat{\tau}_{\Sigma})^{-1}$. If necessary, we can enforce this by scaling the operators $\hat{\tau}_z$.

6

Empirical Investigations

We have now established a general framework for learning SSs from data that unifies many of the previously published algorithms. To obtain a concrete learning algorithm we need to make specific design decisions, which may involve setting further learning parameters, for each of the learning steps. To recall, these are:

- The selection of indicative and characteristic words (Section 5.2)
- The estimation of a suitable target dimension (Section 5.3)
- The method used to solve the learning Equations (2), where we need to decide if we should employ weights, and if so, which variant of weighted spectral learning to use (Section 5.4).

In this chapter we empirically investigate the effect of the various strategies on the performance of the resulting learning algorithm. Regarding the selection of indicative and characteristic words, we are particularly interested in finding a generic selection strategy, i.e., one that does not rely on parameters that require further tuning and that works well across a diverse range of learning tasks. Furthermore, we are especially interested in investigating the utility of using weights in the spectral learning algorithm, as this is a major contribution of this work.

In Section 6.1 we first introduce several problems and datasets as benchmarks for the learning algorithms. These are selected from the domains of stochastic processes and controlled processes as problems that have been used previously as performance benchmarks, but we add two sets of real-world data as well as synthetically generated data sets in an attempt to cover processes of lower and higher complexity, as well as processes with small and large alphabets.

6. EMPIRICAL INVESTIGATIONS

In the subsequent Section 6.2, we investigate several strategies for selecting indicative and characteristic words. At this point, we will use the true process rank as target dimension and therefore consider only the synthetic benchmark data sets. We consider each selection strategy once for the standard spectral learning Algorithm 6 and once for the weighted spectral learning Algorithm 8.

Next, in Section 6.3, we investigate the effect of the target dimension. We again consider two concrete algorithms, namely standard spectral learning as well as weighted spectral learning, where in each case we fix a selection of indicative and characteristic words that performs well, as established in the previous Section 6.2. We then vary the target dimension and thereby establish a range of suitable target dimensions, and based on this, evaluate the methods presented in Section 5.3.1 for estimating a suitable target dimension.

Using the method for selecting indicative and characteristic words as well as the method for estimating a suitable target dimension that we will have established in Sections 6.2 and 6.3, we then compare standard spectral learning to weighted spectral learning and the variants of weighted spectral learning that we have presented in Sections 5.4.3, 5.4.4 and 5.4.5. At this point we will also include the real-world data sets.

6.1 Benchmark Problems and Datasets

In selecting problems and datasets to evaluate the performance of the learning algorithms, we restrict ourselves to consider what we regard as typical application scenarios from the domains of stochastic processes and controlled stochastic processes. We exclude the domain of stochastic languages (as well as other domains where SSs may potentially be applicable) for several reasons. Foremost, we have expertise and implementations for learning SSs for (controlled) stochastic processes, and while SSs for stochastic languages are closely related to SSs for stochastic processes, the implementation details differ considerably. Furthermore, datasets and comparison results are available for the case of (controlled) stochastic processes but not for stochastic languages. And finally, we are interested in evaluating the impact of various design choices on the performance of the resulting learning algorithms and can obtain sufficiently meaningful results by restricting ourselves to (controlled) stochastic processes.

6.1.1 Benchmarks for Controlled Processes

SSs for controlled processes have been investigated in the form of PSRs primarily on a set of POMDP benchmark tasks of varying complexity from Cassandra’s POMDP repository [Cassandra, 1999]. Extensions of PSRs to the case of continuous-valued processes have used quite different datasets, but these are no longer SSs and are beyond the scope of this thesis. Here we will employ the same set of POMDPs that have been used in previous work and for which comparison data is therefore available [Singh et al., 2003; James and Singh, 2004; Wolfe et al., 2005; McCracken and Bowling, 2006]. We are not concerned with the details of the individual problems but instead summarize the main features of these POMDP tasks in Table 4.

POMDP	$ \Sigma_I $	$ \Sigma_O $	dim	rank	$x_{\min}$	$y_{\min}$
TIGER	3	2	2	2	1	1
PAINT	4	2	4	2	1	1
BRIDGE	12	5	5	5	1	1
NETWORK	4	2	7	7	1	2
SHUTTLE	3	5	8	7	3	1
MAZE4X3	4	6	11	10	1	2
CHEESE	4	7	11	11	3	2

Table 4: Properties of the POMDP benchmark problems. $|\Sigma_I|$ and $|\Sigma_O|$ are the sizes of the input and output alphabets, respectively. “dim” denotes the number of POMDP states, which equals the dimension of the converted (but not minimized) SS, while “rank” denotes the rank of the POMDP as a SS. The quantities $x_{\min}$ and $y_{\min}$ are the minimum required length of at least one indicative or characteristic sequence, respectively, such that the system matrix $F_{Y,X}$ has full rank.

One should note that POMDPs are usually used as world models for planning tasks in the context of reinforcement learning, i.e., feedback rewards are an additional part of the POMDP model. Here, however, we only consider the POMDP as a model for observations given actions and therefore disregard the reward signal, even though the learned SS for this controlled process will be unsuitable for planning. This is in line with how POMDPs have been used previously as benchmarks for PSR learning. It is possible to learn a complete world model including rewards using SSs and to use this for planning [James et al., 2004; James and Singh, 2005; Even-Dar et al., 2005; Izadi and Precup, 2008; Boots et al., 2010], but this goes beyond the basic SS learning task that we consider here.

Next, we wish to sample training and test sequences from these POMDPs, but for this we need to specify a policy, i.e., how the input actions are generated. In previous

6. EMPIRICAL INVESTIGATIONS

PSR learning benchmarks a uniform iid policy was used, selecting each action with equal probability, which corresponds to fully random exploration. We will therefore also employ this random policy.

For each POMDP, we generate 3 training sequences of length 10^7 and 5 test sequences of length 10^4 . We will generally measure the learning performance by the average performance of a learned model on the 5 test sequences, which we average over 3 training trials using the 3 training sequences or shorter prefixes thereof. We measure the performance of a model on a test sequence $\bar{x}$ as the *average one-step prediction error* defined as follows. This is the same measure that has been used in the PSR literature.

Definition 61. *Given a SS $\hat{\mathcal{M}}$ for a (controlled) process, as well as a sequence $\bar{x} = a_1 o_1 \dots a_n o_n$ and a “correct” model $\mathcal{M}$, the average one-step prediction error of $\hat{\mathcal{M}}$ with respect to $\mathcal{M}$ and $\bar{x}$, denoted $e(\hat{\mathcal{M}} | \mathcal{M}, \bar{x})$ is defined as*

$$e(\hat{\mathcal{M}} | \mathcal{M}, \bar{x}) = \frac{1}{|\bar{x}|} \sum_{t=1}^{|\bar{x}|} \frac{1}{|\Sigma_O|} \sum_{o \in \Sigma_O} \left(\dot{f}_{\hat{\mathcal{M}}}(a_t o_t | \bar{h}_t a_t) - \mathbb{P}(o_t | \bar{h}_t a_t) \right)^2,$$

where $\bar{h}_t = a_1 o_1 \dots a_{t-1} o_{t-1}$ is the history at time t , $\mathbb{P}(o_t | \bar{h}_t a_t) = f_{\mathcal{M}}(\bar{h}_t a_t o_t) / f_{\mathcal{M}}(\bar{h}_t)$ is the probability of observing o_t at time t given history $\bar{h}_t$ and action a_t according to the “correct” model $\mathcal{M}$, and $\dot{f}_{\hat{\mathcal{M}}}(a_t o_t | \bar{h}_t) = \dot{f}_{\hat{\mathcal{M}}}(\bar{h}_t a_t o_t) / \dot{f}_{\hat{\mathcal{M}}}(\bar{h}_t)$ is the corresponding probability according to the normalized model estimate $\hat{\mathcal{M}}$ (cf. Section 5.6).

Note that this error measure will be zero if $\hat{\mathcal{M}} \cong \mathcal{M}$, i.e., for a perfect model estimate.

6.1.2 Benchmarks for Stochastic Processes

Common benchmarks for studying SS learning of stochastic processes are not available. However, previous OOM research has studied simple English text data or synthetic data sampled from randomly generated OOMs [Jaeger et al., 2006a,b]. Here, we consider gene sequence data in addition to English text data in the form of two real-world datasets from the large Canterbury Corpus [Bell et al., 1997], which is a benchmark for lossless compression. We complement this with synthetically generated data from randomly generated OOMs to cover processes of lower and higher complexity, as well as processes with small and large alphabets. However, we will take more care than in previous work to ensure that the random OOM generation process yields stochastic processes relevant for measuring the learning performance.

6.1 Benchmark Problems and Datasets

Each benchmark will consist of one or more training sequences of length $\sim 10^7$ and 5 test sequences of length $\sim 10^4$. We measure the learning performance by the average performance of a learned model on the test sequences and measure the performance of a model on a test sequence $\bar{x}$ by its *time-averaged negative log-likelihood*.

Definition 62. Given a SS $\hat{\mathcal{M}}$ for a stochastic process and a sequence $\bar{x}$, the time-averaged negative log-likelihood of $\hat{\mathcal{M}}$ given $\bar{x}$, denoted by $\ell(\hat{\mathcal{M}}|\bar{x})$, is defined as

$$\ell(\hat{\mathcal{M}}|\bar{x}) = -\frac{1}{|\bar{x}|} \log_2 \left(\dot{f}_{\hat{\mathcal{M}}}(\bar{x}) \right),$$

where $\dot{f}_{\hat{\mathcal{M}}}(\bar{x})$ is the probability of $\bar{x}$ according to the normalized model estimate $\hat{\mathcal{M}}$ (cf. Section 5.6).

Note that finding a model $\hat{\mathcal{M}}$ that minimizes $\ell(\hat{\mathcal{M}}|\bar{x})$ on the training sequence $\bar{x}$ would correspond to a maximum likelihood estimate, but we measure the performance on test sequences instead to measure the generalization performance of the learned model $\hat{\mathcal{M}}$. Furthermore, if the training and test data are sampled from a stationary ergodic process described by an (unknown) function f , then this measure can be seen as an estimate of the *cross-entropy rate* $H(f, \dot{f}_{\hat{\mathcal{M}}})$ of the process described by f with respect to the process described by $\dot{f}_{\hat{\mathcal{M}}}$ by the Shannon-McMillan-Breiman theorem [Breiman et al., 1957]. This view allows a nice interpretation, as the cross-entropy rate $H(f, \dot{f}_{\hat{\mathcal{M}}})$ is the average number of bits per time step needed to encode a sample from the true data distribution using an optimal coding based on the normalized learned model $\hat{\mathcal{M}}$. Clearly, the smaller the cross-entropy rate, the better $\hat{\mathcal{M}}$ describes the true data distribution. This is a well-known link between estimation theory from the field of statistics and coding theory from the field of information theory.

Furthermore, $H(f, \dot{f}_{\hat{\mathcal{M}}}) \geq H(f, f) = H(f)$, where $H(f)$ is the *entropy rate* of the process described by f , which says that we cannot find a better model for the process described by f than f itself. This means that we can view $H(f)$ as a reasonable target lower bound for the performance measure $\ell(\hat{\mathcal{M}}|\bar{x})$. However, this is not exact, as $\ell(\hat{\mathcal{M}}|\bar{x})$ is only an estimate of $H(f, \dot{f}_{\hat{\mathcal{M}}})$ based on the sample $\bar{x}$. Nevertheless, if we are able to estimate $H(f)$, for instance by $\ell(\mathcal{M}|\bar{x})$ if we have access to an exact model $\mathcal{M}$ for f , we can use this as an indication of how close a learned model $\hat{\mathcal{M}}$ comes to an optimal model for the process described by f .

Finally, we can also give an upper bound for $\ell(\hat{\mathcal{M}}|\bar{x})$ that any reasonable model for the process should satisfy. Namely, consider the process that assigns every symbol the probability $\frac{1}{|\Sigma|}$ independently at each time step, which is described by the function

6. EMPIRICAL INVESTIGATIONS

$g_{|\Sigma|} : \bar{x} \mapsto (\frac{1}{|\Sigma|})^{|\bar{x}|}$. Then $H(g_{|\Sigma|}) = H(f, g_{|\Sigma|}) = \ell(g_{|\Sigma|} | \bar{x}) = \log_2(|\Sigma|)$. Hence, for any learned model $\hat{\mathcal{M}}$ to perform better than the uninformed model $g_{|\Sigma|}$, it should satisfy $\ell(\hat{\mathcal{M}} | \bar{x}) < \log_2(|\Sigma|)$. This is not exact, as $\ell(\hat{\mathcal{M}} | \bar{x})$ is only an estimate of $H(f, \hat{f}_{\hat{\mathcal{M}}})$.

6.1.2.1 Real-world Data

We first describe the real-world datasets that we will be using. The English text is the King James version of the Bible, consisting of approximately four million symbols from an alphabet of 26 uppercase and 26 lowercase letters (while the ‘X’ does not occur), 10 punctuation symbols (‘.’, ‘!’, ‘?’, ‘,’, ‘;’, ‘:’, ‘(’, ‘)’, ‘-’, ‘"’), and two white-space symbols, giving an alphabet size of 64. We preprocess this dataset to yield different sets of training and test sequences. First, we split the data into sentences according to the punctuation symbols ‘.’, ‘?’ and ‘!’, which gives 28,926 sentences, which we shuffle. We then replace all punctuation symbols by whitespace, reduce all whitespace occurrences to a single space symbol, remove any whitespace at the beginning of sentences, and convert all letters to uppercase. Next, we randomly distribute the sentences into a set of training sentences and five sets of test sentences, such that each set of test sentences consists of approximately 10,000 symbols. We concatenate the training sentences to obtain the training sequence and do the same for each set of test sentences to obtain five test sequences. The point of the sentence shuffling is to ensure that the test sequences and training sequence have similar statistical properties. This procedure yields our first benchmark data set that we denote by BIBLE, which has an alphabet size of 27 (26 letters plus one whitespace symbol).

The gene sequence data is a complete genetic sequence of the *E. Coli* bacterium, which is a circular sequence of approximately 4.6 million nucleotides denoted by the symbols ‘G’, ‘A’, ‘C’ and ‘T’. As we have no insight into the structure of the data, we create five test sequences by cutting five sequences of length 10,000 at equally spaced intervals from the original sequence and concatenate the remaining pieces to give the training sequence. We refer to the resulting benchmark data as ECOLI. Note that this creates four positions in the training sequence where preceding and following sequences may not normally occur together, potentially introducing small errors into the estimates $\hat{f}(\bar{x})$ – an issue that we disregard.

This establishes the real-world data that we will be using to benchmark the performance of the SS learning algorithms for stochastic processes (see Table 5 for a summary) and covers processes with different alphabet sizes. However, we do not know much about the complexity of the data or the rank of the underlying generator. We

6.1 Benchmark Problems and Datasets

therefore complement our benchmarks by synthetically generated data.

DATASET	$ \Sigma $	rank	training sequences		test sequences	
			#	length	#	length
BIBLE	27	?	1	3,846,968	5	~10,000
ECOLI	4	?	1	4,588,690	5	10,000

Table 5: Summary of the real-world benchmarks for SS learning of stochastic processes.

6.1.2.2 Synthetic Data

We will create four additional datasets from randomly created OOMs with smaller and larger alphabet sizes and ranks. For this we choose alphabet sizes of 4 and 27 to match the alphabet sizes of the real-world data sets, as well as ranks of 7 and 32, which should resemble processes of lower and higher complexity. It is hard to justify the choice of precisely these numbers, except that using a power of 2 for the larger rank slightly improves computational efficiency. We will denote the resulting datasets as RANDOM4_7, RANDOM4_32, RANDOM27_7 and RANDOM27_32. However, we need to specify how to create OOMs randomly.

To generate a random d -dimensional OOM for an alphabet of size $|\Sigma|$, we preset $\sigma = \mathbb{1}_d^\top$. We then draw each entry of the matrices τ_z independently from a uniform distribution on the interval $[0, 1]$, raise each entry of τ_z to the power α , where α is a control parameter, and rescale the columns of the matrices τ_z such that $\sigma\tau_\Sigma = \sigma$, i.e., such that the matrix $\tau_\Sigma = \sum_{z \in \Sigma} \tau_z$ has column sums of one. Next, we set all entries below a threshold ε_0 to zero and again rescale the columns of the matrices τ_z such that $\sigma\tau_\Sigma = \sigma$. Finally, we compute ω_ε as the *stationary state*, i.e., the unique eigenvector of τ_Σ to the eigenvalue 1 that satisfies $\sigma\omega_\varepsilon = 1$ [Jaeger et al., 2006a]. Note that the existence of a unique stationary state ensures that the resulting process is stationary and ergodic [Schönhuth and Jaeger, 2009]. If such a unique stationary state does not exist, the procedure is repeated. The parameter α controls the relative sizes of the entries within the columns of the matrices τ_z . Setting $\alpha = 0$ will enforce all entries to be $1/|\Sigma|$, corresponding to a process that produces each symbol with probability $1/|\Sigma|$ independently at each time step. Higher settings of α cause many of the entries to become very small due to the rescaling of the columns and therefore enforces some form of sparsity on the matrices τ_z , which is made precise by setting all entries smaller than ε_0 to zero. The choice of ε_0 should reflect the threshold below which we consider an entry to have negligible effect on the model. We use a fixed setting of $\varepsilon_0 = 10^{-7}$. The

6. EMPIRICAL INVESTIGATIONS

effect of α on the resulting process has never been studied, but values for α between one and ten have typically been used. However, we will argue that the resulting processes are not well-suited as benchmarks, and we will use much higher settings of α .

Our goal is to generate stochastic processes that exhibit a sufficiently rich structure. To make this precise, we will require such a process described by f to have the following two properties:

- (i) $\min\{H(f, q) - H(f) : q \text{ is a 3-rd order Markov process}\}$ should be as large as possible, so that the process differs from a 3-rd order Markov process.
- (ii) The entropy $H(f)$ should lie well within the interval $[0, \log_2(|\Sigma|)]$.

The first property ensures that the learning task is non-trivial, and the second property ensures that we have a large enough margin to measure the performance of reasonable models between the performance bounds $\log_2(|\Sigma|)$ and $H(f)$ discussed above, while at the same time having a non-deterministic process. Note that for a stationary and ergodic process described by f , the k -th order Markov process q_k that minimizes $H(f, q_k)$ is given by the matrix $F_{Y,X}$ after normalizing to have column sums of 1, where $X = \Sigma^k$ and $Y = \Sigma$.

We will therefore use the following approach: For different values of α , d , and $|\Sigma|$, we generate 50 d -dimensional OOMs each randomly as described above and select the model $\mathcal{M}$ that maximizes the relative entropy $\hat{H}(f_{\mathcal{M}}, q) - \hat{H}(f_{\mathcal{M}})$ for an optimal 3-rd order Markov model q derived from $\mathcal{M}$, with estimates $\hat{H}(f_{\mathcal{M}}, q)$ and $\hat{H}(f_{\mathcal{M}})$ computed on a sequence of length 10^5 sampled from $f_{\mathcal{M}}$. We repeat this three times and from the three selected models choose the one with smallest entropy $\hat{H}(f_{\mathcal{M}})$ that we estimate more precisely on sample sequences of length 10^7 . The results are shown in Figure 2.

What we see is that small values of α lead to OOMs describing stochastic processes that are almost purely random, i.e., uniform iid processes, or that are well-described by 3-rd order Markov models. Note that this is the case even for OOMs that are already pre-selected to be far from 3-rd order Markov processes. Furthermore, for larger dimensional OOMs, we need to select larger values of α to obtain “interesting” processes. Finally, low-dimensional OOMs with large alphabets generated in this way for any choice of α lead to stochastic processes that are described well by 3-rd order Markov models.

As benchmarks, we select the 7-dimensional OOMs corresponding to the value of $\alpha = 40$ and the 32-dimensional OOMs corresponding to the value of $\alpha = 160$. For each of the four OOMs that we obtain in this way, we generate three training sequences of length 10^7 (or in fact of length 10^8 for the 32-dimensional OOM with alphabet size 27)

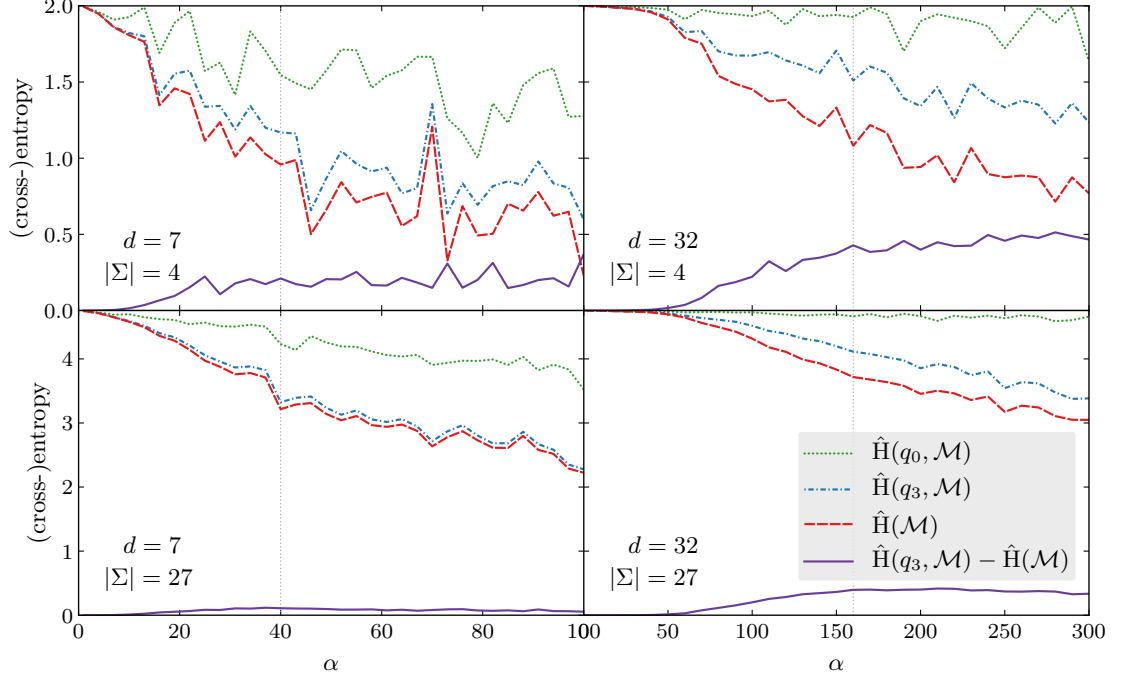


Figure 2: For d -dimensional OOMs $\mathcal{M}$ with alphabet Σ generated and selected as described above for various values of d , $|\Sigma|$ and α , the estimated entropy rate $\hat{H}(f_{\mathcal{M}})$ and the estimated cross-entropy rates $\hat{H}(f_{\mathcal{M}}, q_0)$ and $\hat{H}(f_{\mathcal{M}}, q_3)$ of best-approximating 0-th and 3-rd order Markov models are shown, as well as the difference $\hat{H}(f_{\mathcal{M}}, q_3) - \hat{H}(f_{\mathcal{M}})$. The entropies are estimated by $\ell(\mathcal{M} | \bar{x})$, $\ell(q_0 | \bar{x})$ and $\ell(q_3 | \bar{x})$, respectively, on a sequence $\bar{x}$ of length 10^7 sampled from $\mathcal{M}$.

and five test sequences of length 10^5 that form our benchmark data sets RANDOM4_7, RANDOM4_32, RANDOM27_7 and RANDOM27_32. A summary is given in Table 6.

DATASET	$ \Sigma $	rank	sparsity	$\log_2(\Sigma)$	$\hat{H}(f, q_0)$	$\hat{H}(f, q_3)$	$\hat{H}(f)$
RANDOM4_7	4	7	0.66	2	1.54532	1.16933	0.95743
RANDOM4_32	4	32	0.90	2	1.92694	1.51314	1.08093
RANDOM27_7	27	7	0.70	4.755	4.23171	3.32650	3.21578
RANDOM27_32	27	32	0.92	4.755	4.66523	4.11195	3.71746

Table 6: Summary of the synthetic benchmarks for SS learning of stochastic processes. Sparsity measures the fraction of entries in the matrices τ_z that are zero. $\hat{H}(f, q_0)$ and $\hat{H}(f, q_3)$ are the estimated cross-entropy rates for best-approximating 0-th and 3-rd order Markov processes q_0 and q_3 , and $\hat{H}(f)$ is the estimated entropy rate of the underlying process described by f .

6. EMPIRICAL INVESTIGATIONS

6.1.3 Default Evaluation on Benchmarks

In the following we will be evaluating and comparing several algorithms that we obtain by varying the approaches for the individual learning steps. Since we will be conveying a lot of information in the comparison graphs, which, however, all have the same evaluation format, we now explain in more detail how the evaluation of a single algorithm is performed and presented.

Let us consider a standard spectral learning algorithm as in Algorithm 6, where we select as indicative and characteristic words all words of length less than or equal to L that occur at least once in the training data, where the length L is preset for each benchmark to the largest integer such that $|\Sigma|^L \leq 1024$, which gives reasonable results. We use the process rank as target dimension if available, or, in the case of the real-world benchmarks, estimate a suitable target dimension using a method from Section 5.3.

For each benchmark we are given one or three training sequences. For each training sequence, we consider prefixes of increasing lengths that we use as data for the model estimation. The lengths we use are $\{\lfloor 10^{\frac{k}{2}} \rfloor \mid k = 6, 7, \dots, 14\}$ covering a range of lengths from 10^3 to 10^7 , except for the RANDOM4_7 benchmark, where we instead use lengths ranging from 10^2 to 10^6 , the RANDOM27_32 benchmark, where we use lengths ranging from 10^4 to 10^8 , and the real-world benchmarks, where we use lengths ranging from 10^3 to about $4 \cdot 10^6$, the length of the training sequence. We then estimate a model using the specified learning algorithm for each such training sequence prefix and evaluate it on the five test sequences for the respective benchmark using the evaluation metric we have explained above, i.e., in the case of stochastic processes, we compute the time-averaged negative log-likelihood of the estimated model given the test sequence, and in the case of controlled processes, we compute the average one-step prediction error. We take the arithmetic average over the evaluations for the five test sequences, and if we have three training sequence prefixes of every length, as is the case for all synthetic benchmarks, we take the arithmetic average of their evaluation results as well.

Note that it is essential to employ some form of normalization when using the likelihood to evaluate the model, as it may happen that the learned model gives a probability of zero (or less) to a test sequence, for instance when a symbol appears that was not present in the training data, which renders the likelihood meaningless. This occurs also when learning models such as HMMs that are guaranteed to be (controlled) stochastic processes. For our evaluations we want to use only mild normalization and stabilization to avoid confounding effects, and we will use the following normalization

6.1 Benchmark Problems and Datasets

and stabilization settings throughout: $p_{\min} = 0.0002$, $e_{\max} = 0.03$, $l_{\max} = 5$, $\varepsilon_0 = 10^{-8}$ for the benchmarks for stochastic processes, and $p_{\min} = 0$, $e_{\max} = 1$, $l_{\max} = 0$, $\varepsilon_0 = 10^{-8}$ for the benchmarks for controlled stochastic processes (cf. Section 5.6). We will not consider optimizing the normalization and stabilization settings, since this is highly dependent on the desired application of the learned model. While these settings do affect our performance measures, we have observed that these generally do not favor one algorithm over another, i.e., the *relative* performance of the algorithms we consider is not affected significantly. This appears to hold as long as we employ sufficient stabilization (i.e., not too large $e_{\max}$) to avoid obvious stability issues. Using the above pre-determined settings should, therefore, be sufficient for our comparison purposes.

We first consider the benchmarks for controlled processes as well as the synthetic benchmarks for stochastic processes, since in these cases we know the correct model and therefore also the entropy and rank, and we will use only these benchmarks to evaluate the methods for selecting indicative and characteristic words as well as the target dimension in the following Sections 6.2 and 6.3. The evaluation results on these benchmarks for the single algorithm described above are presented in Figure 3. The evaluation results on the real-world benchmarks for stochastic processes, which will be used when comparing the variants of (weighted) spectral learning in Section 6.4, are shown in Figure 4. In all cases, the x -axis shows the length of training data used in logarithmic scale, and the y -axis shows the averaged evaluation on the test data. The average one-step prediction error that is used in the case of controlled processes is shown in logarithmic scale with a fixed range from 10^{-6} to 10^{-1} , while the negative log-likelihood is shown on a linear scale ranging, in the case of the synthetic benchmarks for stochastic processes, from $\hat{H}(f_{\mathcal{M}})$ to $\hat{H}(f_{\mathcal{M}}, q_0)$, where $\mathcal{M}$ is the respective benchmark model and q_0 is the best 0-th order Markov model approximation to $f_{\mathcal{M}}$. This means that the performance measure in the case of synthetic benchmarks for stochastic processes is expected to converge to the x -axis as the length of training data increases, while in the case of the benchmarks for controlled processes it is expected to keep decreasing. Due to the chosen range for the negative log-likelihood, the performance measure can also be interpreted as a plot of the relative entropy of the estimated model, which decreases to zero for a perfect model. In the case of the real-world benchmarks, we do not know the process entropy and also no best 0-th order Markov approximation to the process, so we simply select a reasonable range for the performance measure on the y -axis, which means that the lower bound has no particular meaning in this case. We will use the same axis scaling throughout and therefore omit the axis labeling from all future evaluation plots.

6. EMPIRICAL INVESTIGATIONS

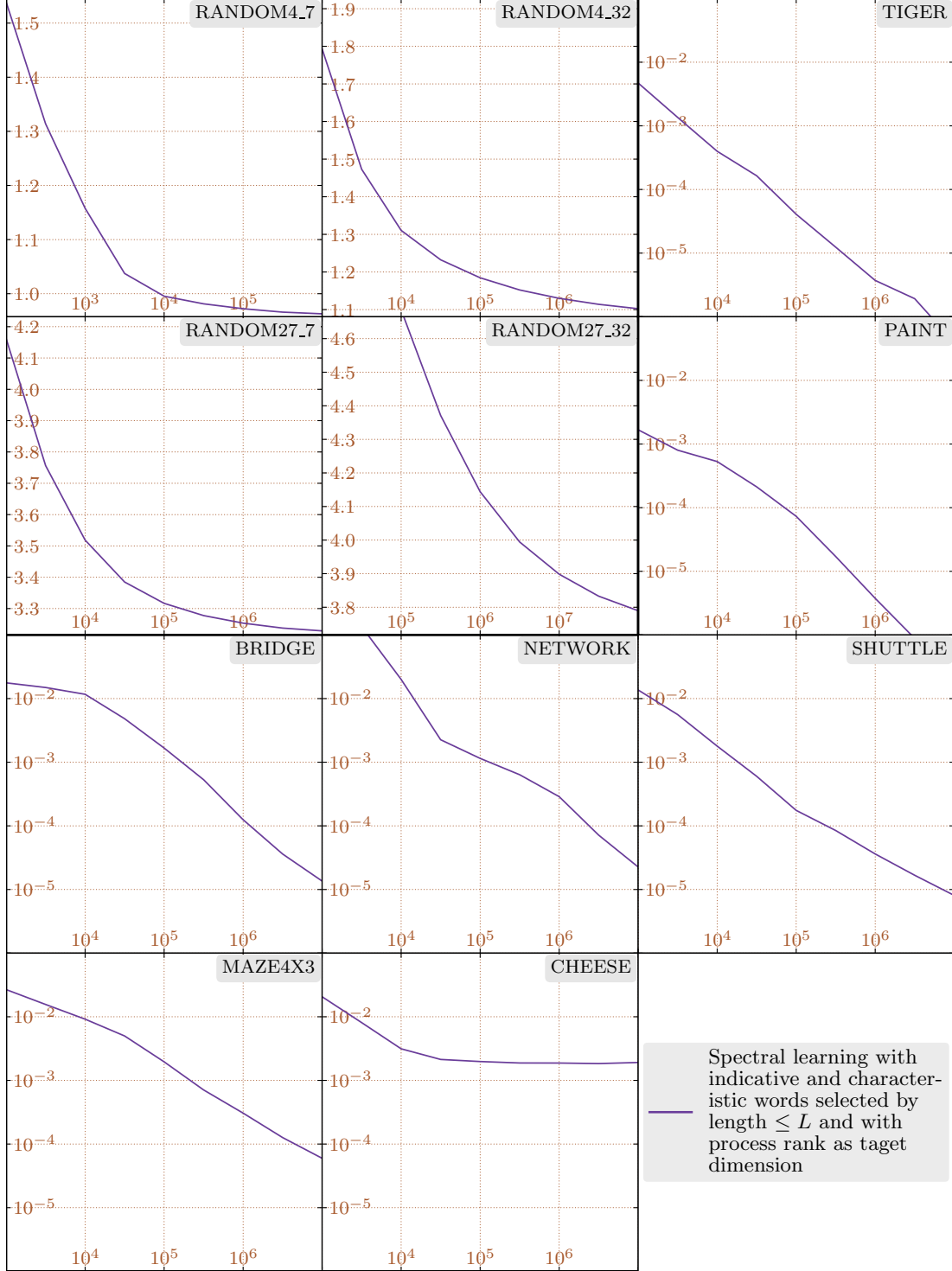


Figure 3: Evaluation of a standard spectral learning algorithm on the synthetic benchmarks for (controlled) stochastic processes.

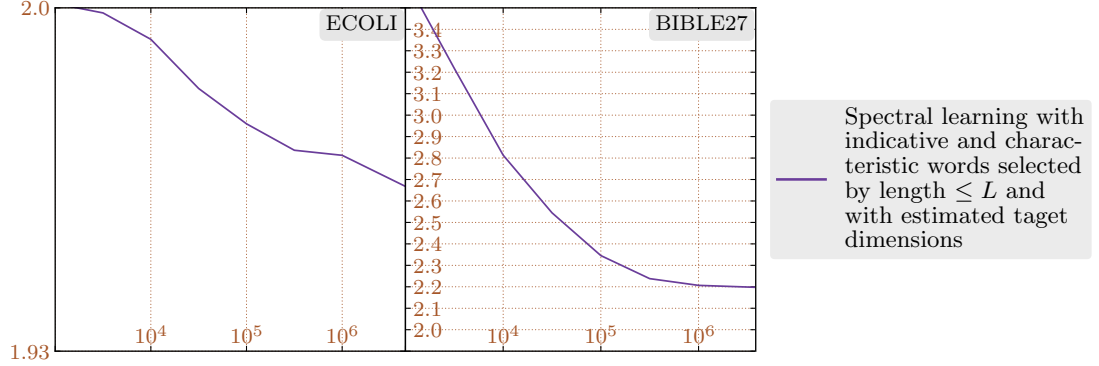


Figure 4: Evaluation of a standard spectral learning algorithm on the real-world benchmarks for stochastic processes.

6.2 Selection of Indicative and Characteristic Words

We begin by evaluating various approaches to selecting indicative and characteristic words based on the general selection procedure from Section 5.2.1. For this, we will use the process rank as target dimension, assuming that this will not change the outcome qualitatively, i.e, we disregard any interaction between the selection of indicative and characteristic words and the choice of target dimension. We will see in the next section that choosing the process rank as target dimension indeed often gives very reasonable results. However, there are conceivable issues, for instance, using a large matrix $\hat{F}_{Y,X}$ combined with a large target dimension in the case of limited training data may be especially prone to overfitting, which means we may be introducing a bias towards a selection of smaller sets of indicative and characteristic words in the case of limited training data, but we ignore such possible issues here.

We consider two algorithms, namely the standard spectral learning Algorithm 6, as well as the weighted spectral learning Algorithm 8. The reason is that we conjecture that weighted spectral learning may be much better suited to a selection of indicative and characteristic words of varying lengths than standard spectral learning.

Note that for performance reasons we need to set a limit on the size of the matrix $\hat{F}_{Y,X}$. We will, therefore, only allow sets X, Y of indicative and characteristic words of size at most 1024. Even much larger matrices can be used, but these require special linear algebra routines, for instance handling of large sparse matrices and truncated SVD computations thereof that we have not implemented.

First, we consider a selection of indicative and characteristic words consisting of all words of length L that occur in the training data, for parameters $L_{\min} \leq L \leq L_{\max}$

6. EMPIRICAL INVESTIGATIONS

that need to be specified in some way. This has been the most common way to select indicative and characteristic words. For each benchmark, we will consider all values of $L_{\max}$ such that the resulting matrix is not too large, for which we require that $|\Sigma|^{L_{\max}} \leq 1024$. In this case we do not restrict the size of X and Y via the parameter $Z_{\max}$. For each admissible value of $L_{\max}$ we consider $L_{\min} = L_{\max}$, which means we select only words of length $L_{\max}$, as well as $L_{\min} = 0$, which means we select words of length less than or equal to $L_{\max}$. When using $L_{\min} = 0$, the size of X and Y may, in fact, be slightly larger than 1024. For each benchmark, the evaluation curves corresponding to the largest admissible $L_{\max}$ are plotted slightly thicker for better recognition.

Furthermore, we consider using the $Z_{\max}$ words occurring most frequently in the training data as indicative and characteristic words, which again requires specifying a parameter $Z_{\max}$ that controls the size of the resulting matrix $\hat{F}_{Y,X}$. For all benchmarks, we consider all $Z_{\max}$ in $\{32, 64, 128, 256, 512, 1024\}$, and plot the evaluation curves corresponding to $Z_{\max} = 1024$ slightly thicker for better recognition.

Finally, we consider a selection of the at most 1024 most frequent words that occur at least $O_{\min}$ times in the training data, where $O_{\min}$ is computed as we have proposed in Section 5.2.1. Note that the value of $O_{\min}$ depends on the alphabet size as well as the length of available training data and can, therefore, be seen as a tuned version of the selection based on the $Z_{\max}$ most frequent words. We are particularly interested in the performance of this approach, since it is a novel contribution that obviates the requirement to specify any further parameters.

Note that for simplicity we will not consider the further restriction on the selection of words that is controlled by the parameter U_{unique} . In our experiments, this has shown only little impact on the performance of the learning algorithms.

The evaluations for these approaches are shown in Figure 5 when combined with standard spectral learning and in Figure 6 when combined with weighted spectral learning. To facilitate comparison of results across these two figures, we have included in the first Figure 5 the evaluation for our novel selection approach based on a heuristically set parameter $O_{\min}$ combined with weighted spectral learning, and in the second Figure 6 the evaluation for the selection based on all words of length less than or equal L combined with standard spectral learning for the largest L such that $|\Sigma|^L \leq 1024$.

Let us focus first on the evaluations for indicative and characteristic words selected based on a fixed length L or a length $\leq L$. We can see that using a too small length L such that $|\Sigma|^L < d$, where d is the process rank, prevents convergence to a correct model as the training data increases. This is clearly seen for instance for $L = 2$ in the

6.2 Selection of Indicative and Characteristic Words

RANDOM4.32 case. The reason for this is that the columns (or rows) of F indexed by the indicative words X (or characteristic words Y) cannot span the column (or row) space of F , i.e., the sets X and Y cannot be sufficient. To guarantee sufficiency, we would need to choose $L \geq d$. Interestingly, as has already been observed before, a choice of L such that $|\Sigma|^L \gg d$ usually does seem to give sufficient sets X and Y and thereby achieve convergence to a correct model. However, we do observe an exception to this for the CHEESE benchmark, where considering the $28^2 = 784$ words of length 2 (or even ≤ 2) as indicative and characteristic words, of which in this case 166 also occur in the largest available training data, is not sufficient to estimate a correct model of rank 11. Furthermore, we can easily construct cases where using words of length less than or equal to L , where L is chosen as large as possible while satisfying $|\Sigma|^L \leq M$ for some fixed maximum number of words M , is guaranteed to fail. For instance, if we use $M = 1024$ as we do here, then this approach will fail to select sufficient indicative and characteristic words and hence cannot converge to a correct model for any stochastic process with alphabet size $\Sigma > 32$ of rank larger than $|\Sigma| + 1$, since the largest admissible word length is one and hence $\text{rank}(F_{Y,X}) \leq \Sigma + 1$. A more flexible selection method for indicative and characteristic words is thus preferred.

Next, we see that using words of fixed length L and words of variable length $\leq L$ makes little difference on the benchmarks for controlled processes. However, on the benchmarks for stochastic processes, selecting words of variable size $\leq L$ shows a very different performance to words of a fixed length L , and surprisingly this performance is very similar to the performance achieved by selecting the $Z_{\max} = 1024$ most frequent words. Using standard spectral learning, adding the words of length $< L$ to the words of length L clearly improves the performance on the RANDOM4.7 benchmark and clearly hurts the performance on the RANDOM27.32 benchmark, while being inconclusive for the other benchmarks. However, when using weighted spectral learning, the benchmarks RANDOM4.7, RANDOM4.32 and RANDOM27.4 clearly favor adding the words of length $\leq L$, while for the RANDOM27.32 benchmark this depends on the length of training data used.

Finally, selecting all words of length L or $\leq L$ generally gives the best results for the largest L that we consider. However, when selecting only the words of length equal to L , then in some cases (especially RANDOM4.32) it appears that using a smaller length L is advantageous when only limited training data is available. Surprisingly, using weighted spectral learning with a selection of words of length equal to 5 in the TIGER benchmark results in a remarkable degradation of performance compared to all other settings, which we cannot explain.

6. EMPIRICAL INVESTIGATIONS

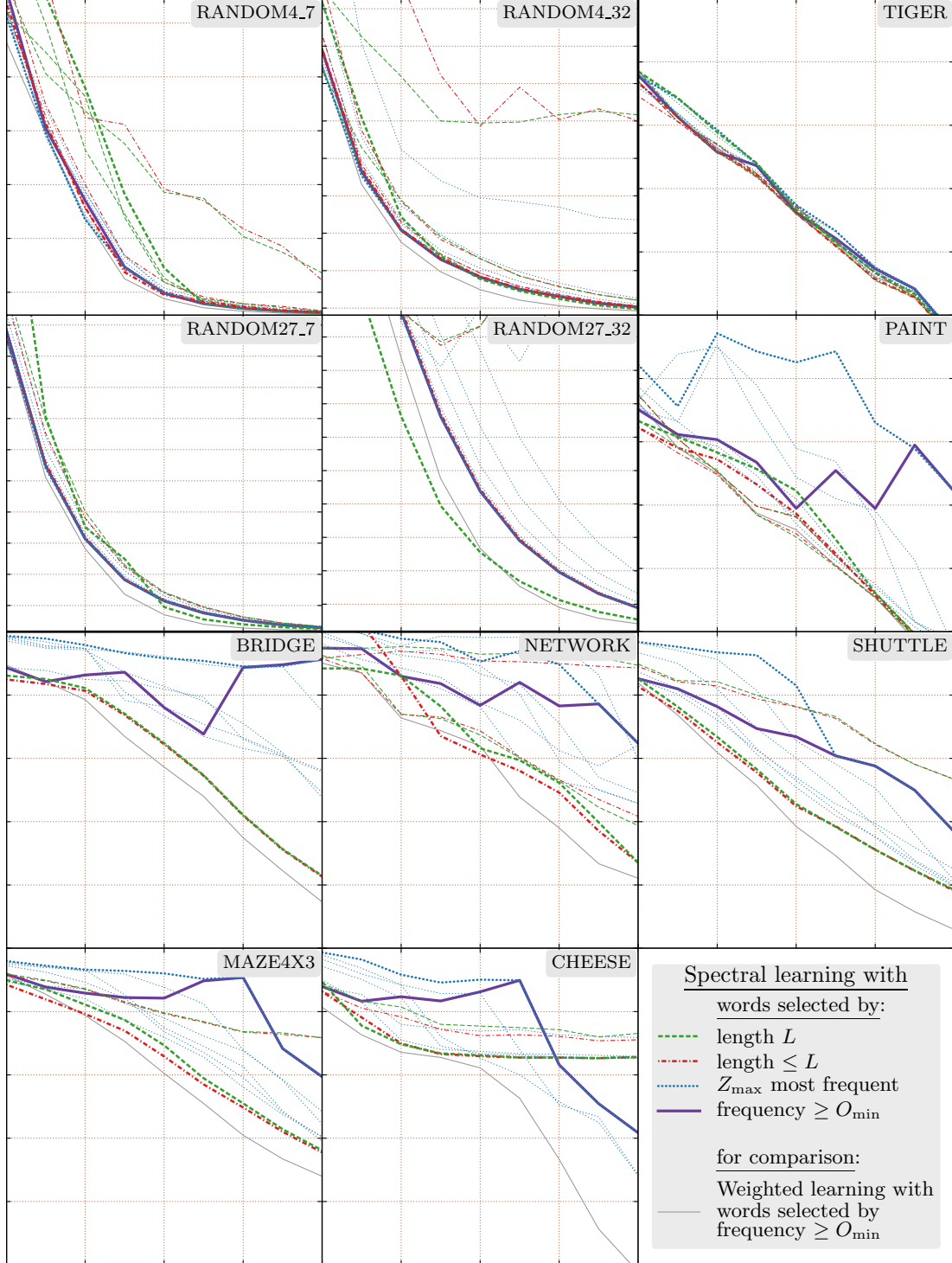


Figure 5: Evaluation of various approaches to selecting indicative and characteristic words, using a standard spectral algorithm and the process rank as target dimension. When selecting by length L or $\leq L$, all L are considered such that $|\Sigma|^L \leq 1024$. When selecting the $Z_{\max}$ most frequent words, all $Z_{\max}$ in $\{32, 64, 128, 256, 512, 1024\}$ are considered. The results for the largest considered L as well as $Z_{\max} = 1024$ are plotted slightly thicker. The selection by frequency $\geq O_{\min}$ uses the formulas presented in Section 5.2.1 to compute a suitable $O_{\min}$.

6.2 Selection of Indicative and Characteristic Words

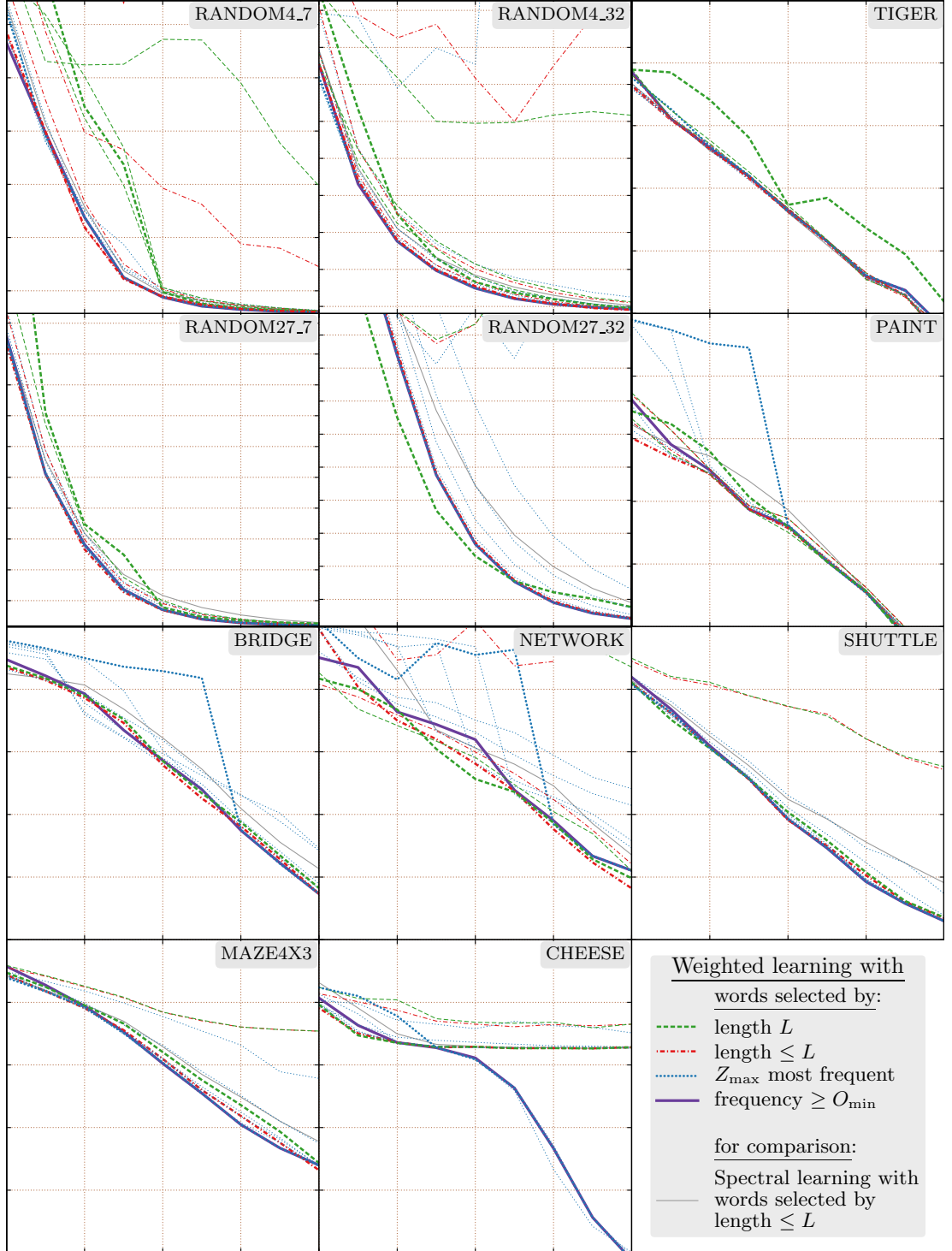


Figure 6: Evaluation of various approaches to selecting indicative and characteristic words, using a weighted spectral algorithm and the process rank as target dimension. When selecting by length L or $\leq L$, all L are considered such that $|\Sigma|^L \leq 1024$. When selecting the $Z_{\max}$ most frequent words, all $Z_{\max}$ in $\{32, 64, 128, 256, 512, 1024\}$ are considered. The results for the largest considered L as well as $Z_{\max} = 1024$ are plotted slightly thicker. The selection by frequency $\geq O_{\min}$ uses the formulas presented in Section 5.2.1 to compute a suitable $O_{\min}$.

6. EMPIRICAL INVESTIGATIONS

Let us summarize our results so far.

Conclusion 63. *Using all words of length less than or equal to L that occur in the training data, while choosing the largest L such that $|\Sigma|^L \leq M$ for some bound M (here $M = 1024$) on the size of the matrix $\hat{F}_{Y,X}$, is a viable approach to selecting indicative and characteristic words. The disadvantage is that this may lead to insufficient sets of indicative and characteristic words (as seen in the CHEESE case). A more flexible approach is therefore preferred.*

Let us now turn to the selection of indicative and characteristic words using the $Z_{\max}$ words occurring most frequently in the training data, which allows for words of varying length. On the benchmarks for stochastic processes, we observe that the largest setting $Z_{\max} = 1024$ gives the best results and that these results are remarkably similar to the performance achieved when selecting indicative and characteristic words by length $\leq L$, when using the largest admissible length L . One possible explanation is that the resulting word selection may be similar, i.e., the most frequent words may happen to coincide with the shortest words, as would be the case if all words of a given length had a similar probability. To investigate, we compute the *overlap* as the percentage of words chosen by length $\leq L$ occurring in the training data that are also among the $Z_{\max} = 1024$ most frequently occurring words, where we compute the average overlap for the training sequences of length 10^7 . This gives 0.16, 0.32, 0.48 and 0.97 for the benchmarks RANDOM4_7, RANDOM4_32, RANDOM27_7 and RANDOM27_32, respectively. We interpret this to indicate that the word selection by the $Z_{\max}$ most frequently occurring words differs significantly from the selection by length $\leq L_{\max}$, and that the overlap in selected words alone is likely not the reason for the striking similarity in performance. Also, things are quite different on the benchmarks for controlled processes. There, using standard spectral learning with a selection of the $Z_{\max}$ most frequent words as indicative and characteristic words performs clearly worse than a selection based on words of length less than or equal to L , with $Z_{\max} = 1024$ giving the worst performance, in some cases (e.g., BRIDGE) apparently not even converging toward a correct model as the training data is increased. Using the proposed refinement requiring a minimum occurrence frequency of $O_{\min}$ that is tuned to the alphabet size and the available training data does not seem to improve the results when using standard spectral learning. The only case where we observe an advantage by this selection approach is the CHEESE benchmark, where the selection of the $Z_{\max}$ most frequent words does lead to sufficient sets of indicative and characteristic words, as we observe convergence to a correct model as the size of the training data increases, provided we

use $Z_{\max} \geq 256$.

However, if we employ weighted spectral learning, the results improve markedly. Using weighted spectral learning, the selection of the $Z_{\max}$ most frequent words as indicative and characteristic words generally gives very good results also on the benchmarks for controlled processes. For sufficiently large training data, using the largest setting $Z_{\max} = 1024$ gives the best results, but, depending on the benchmark, less training data may require a smaller setting of $Z_{\max}$ and therefore a smaller matrix $\hat{F}_{Y,X}$ to achieve optimal performance. This is precisely what the proposed refinement of requiring a minimum occurrence frequency of $O_{\min}$ that is tuned to the alphabet size and the available training data aims to achieve, and in fact this refinement yields a performance that is generally on par with the performance achieved by selecting words by length less than or equal to some L . Furthermore, this refined approach still achieves convergence to a correct model even in the CHEESE benchmark, which means it overcomes the problem that the words selected by length can be too short to be sufficient. Clearly, there is no guarantee that this selection approach cannot run into the same problem, and indeed it does in the CHEESE benchmark if we restrict the maximum number of indicative and characteristic words to $Z_{\max} = 128$. But we conjecture that selecting the words occurring most frequently in the training data, which are therefore in some sense the most representative, does give a high chance of obtaining sufficient sets of indicative and characteristic words that capture all relevant dimensions in $\hat{F}$, but this needs to be made precise and requires further research.

Let us remark on the observation that in many cases the largest matrices we consider give the best results. This might lead us to expect further performance improvements by employing even larger matrices. The selection of sets X, Y of indicative and characteristic words based purely on the minimal occurrence frequency $O_{\min}$ computed by the heuristic presented in Section 5.2.1 would also lead to larger sets X and Y , especially for the synthetic benchmarks for stochastic processes and for large amounts of training data, if we did not restrict their size to 1024 for implementation reasons. However, in the cases where large matrices work best, we see that the next smaller setting of $Z_{\max} = 512$ already achieves similar performance. The results, therefore, suggest that using larger sets of indicative and characteristic words would rather not lead to significant performance improvements but would likely not degrade performance either.

Conclusion 64. *Using the $Z_{\max}$ words occurring most frequently in the training data works reliably only when weighted spectral learning is used. Even so, it is essential to tune the size of the sets of indicative and characteristic words according to the alphabet size and the available training data. The novel heuristic that we have presented in*

6. EMPIRICAL INVESTIGATIONS

Section 5.2.1 achieves this by additionally requiring a minimal occurrence frequency $O_{\min}$ and thereby yields, when combined with weighted spectral learning, the best or close to the best results among all selection strategies that we have considered on these benchmarks. Furthermore, this refined approach appears to generally result in sufficient sets of indicative and characteristic words. The exact upper bound of $Z_{\max} = 1024$ that we have imposed here on the size of the sets of indicative and characteristic does not appear to be critical.

Finally, the comparison curves that we have added to each figure enable us to compare the performance of standard and weighted spectral learning for the two selections of indicative and characteristic words that have shown the most promising results above, namely words of length less than or equal to L occurring in the training data for the largest L satisfying $|\Sigma|^L \leq 1024$, and the at most 1024 most frequent words occurring at least $O_{\min}$ times, with $O_{\min}$ tuned to the alphabet size and available training data via the heuristic presented in Section 5.2.1. For both of these selection strategies, using weighted spectral learning improves the performance in all cases, often significantly. We can also see that the combination of the latter selection strategy with weighted spectral learning performs as good as and in several cases clearly better than *any* standard spectral learning algorithm, the only exception being spectral learning with indicative and characteristic words of length equal to two for the RANDOM27.32 benchmark with limited training data, which leads us to our final conclusion.

Conclusion 65. *On these benchmarks, the combination of weighted spectral learning with indicative and characteristic words selected by requiring a minimal occurrence frequency $O_{\min}$ computed according to the heuristic presented in Section 5.2.1 achieves performance close to or often clearly better than any other algorithm (standard or weighted spectral with various selections of indicative and characteristic words) that we have considered.*

We should note that the general selection procedure introduced in Section 5.2.1 allows further combinations of the selection strategies we have considered here. This is achieved by combining the parameters $L_{\min}$, $L_{\max}$, $Z_{\max}$, $O_{\min}$, and possibly U_{unique} . For instance, we may consider the words of length less than or equal to $L_{\max}$ for some less restrictive bound $L_{\max}$, and instead bound the size of the matrix $\hat{F}_{Y,X}$ by using only the at most $Z_{\max} = 1024$ such words occurring most frequently in the training data. This could be further refined by additionally requiring a minimum occurrence frequency $O_{\min}$ to achieve smaller matrices in the case of limited training data. There appears to be room for further improvements, but at this point we are satisfied to have found a novel

selection strategy that no longer requires tweaking parameters (the crucial parameter $O_{\min}$ is self-tuned via a heuristic), leads to convergence in all considered cases, and performs on par with or better than previous approaches.

6.3 Selecting the Target Dimension

Before comparing the various methods for selecting a suitable target dimension, we first investigate how the choice of target dimension impacts the model estimation. For this, we estimate and evaluate models using various target dimensions, namely $\{1, 2, \dots, d, d+1, d+3, d+10\}$, where d is the process rank, except for the RANDOM4_32 and RANDOM27_32 benchmarks, where the process rank is 32 and we consider only the subset of target dimensions $\{1, 2, 4, 8, 12, 16, 20, 23, 26, 28, 30, 31, 32, 33, 35, 42\}$. We evaluate each target dimension for two algorithms. First, using the standard spectral learning Algorithm 6 with indicative and characteristic words selected as the words of length less than or equal to L occurring in the training data for the largest L such that $|\Sigma|^L \leq 1024$, the results of which are presented in Figure 7. Secondly, using the weighted spectral learning Algorithm 8 with indicative and characteristic words selected as the at most 1024 most frequent words occurring at least $O_{\min}$ times in the training data, where $O_{\min}$ is computed for each alphabet size and amount of training data by the heuristic from Section 5.2.1, the results of which are presented in Figure 8. These algorithms correspond to a standard approach with a reasonable word selection, and to a novel approach using weighted spectral learning and a new word selection approach showing promising results, as discussed in the previous Section 6.2. In both figures we have included “optimality threshold” curves. We will regard any target dimension as suitable for a given training data length if it leads to a performance better than this threshold. The threshold is computed differently for the stochastic and controlled process benchmarks, as they use different performance metrics. In the case of the stochastic process benchmarks, we compute the threshold as the performance of the best model $\hat{\mathcal{M}}$ for the given training data length with an added tolerance of 0.25 times its estimated relative entropy $\hat{H}(f, \hat{f}_{\hat{\mathcal{M}}}) - \hat{H}(f)$, and in the case of the controlled process benchmarks, we compute the threshold as the performance of the best model with an added tolerance of 0.5 times its average one-step prediction error. The tolerance is chosen to capture the notion of being reasonably close to the optimally achievable performance, where we take into account the spread of results that we typically observe when varying other parameters. In fact, the resulting threshold curves correspond roughly to the best performance that can be achieved using only half the available

6. EMPIRICAL INVESTIGATIONS

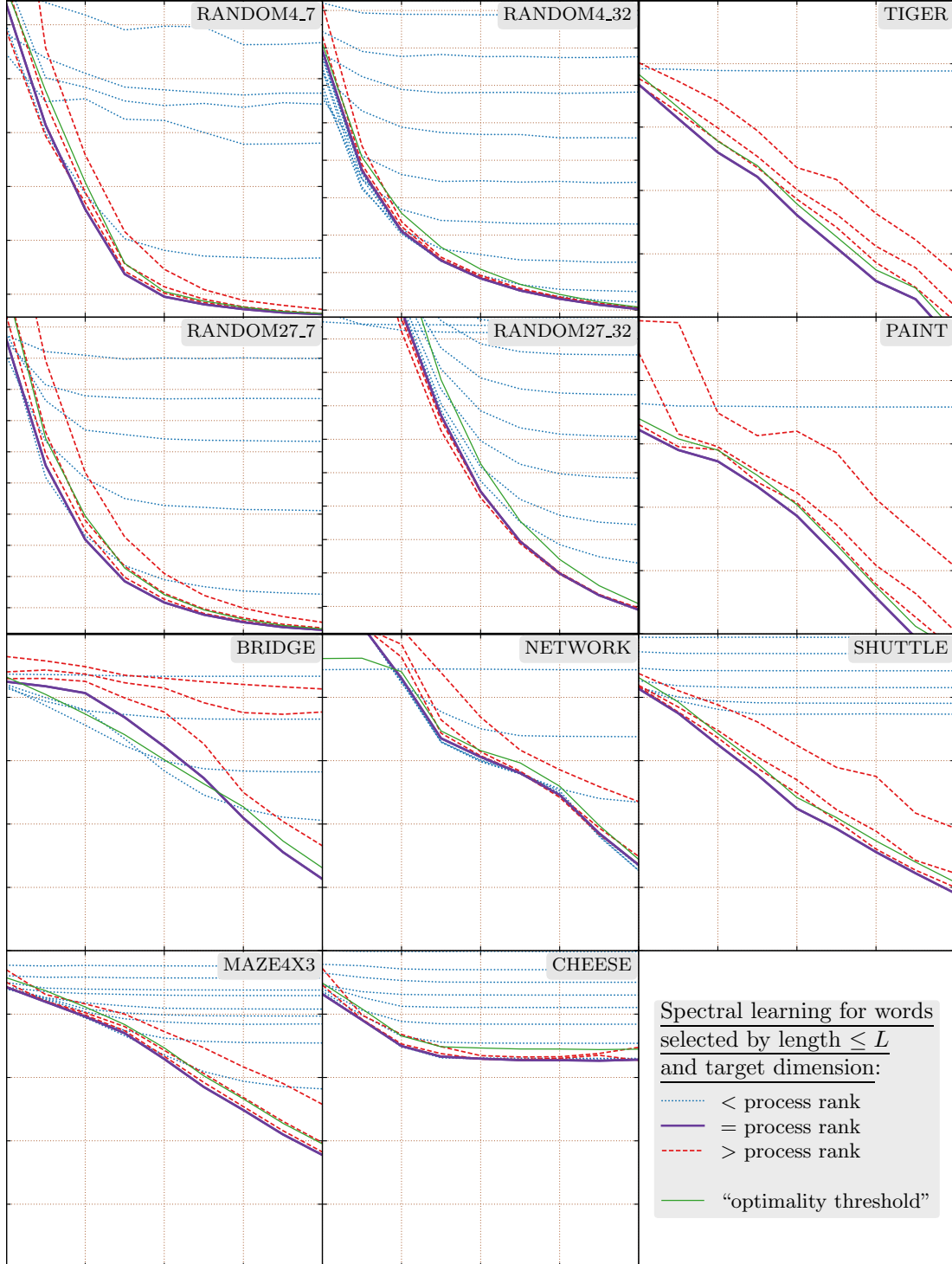


Figure 7: Evaluation of various target dimension settings using the standard spectral algorithm with indicative and characteristic words selected as the words of length less than or equal to L occurring in the training data for the largest L such that $|\Sigma|^L \leq 1024$. For each training data length, the “optimality threshold” is the performance of the best model with an added tolerance of 0.25 times its relative entropy in the case of the stochastic process benchmarks, or else with an added tolerance of 0.5 times its average one-step prediction error.

6.3 Selecting the Target Dimension

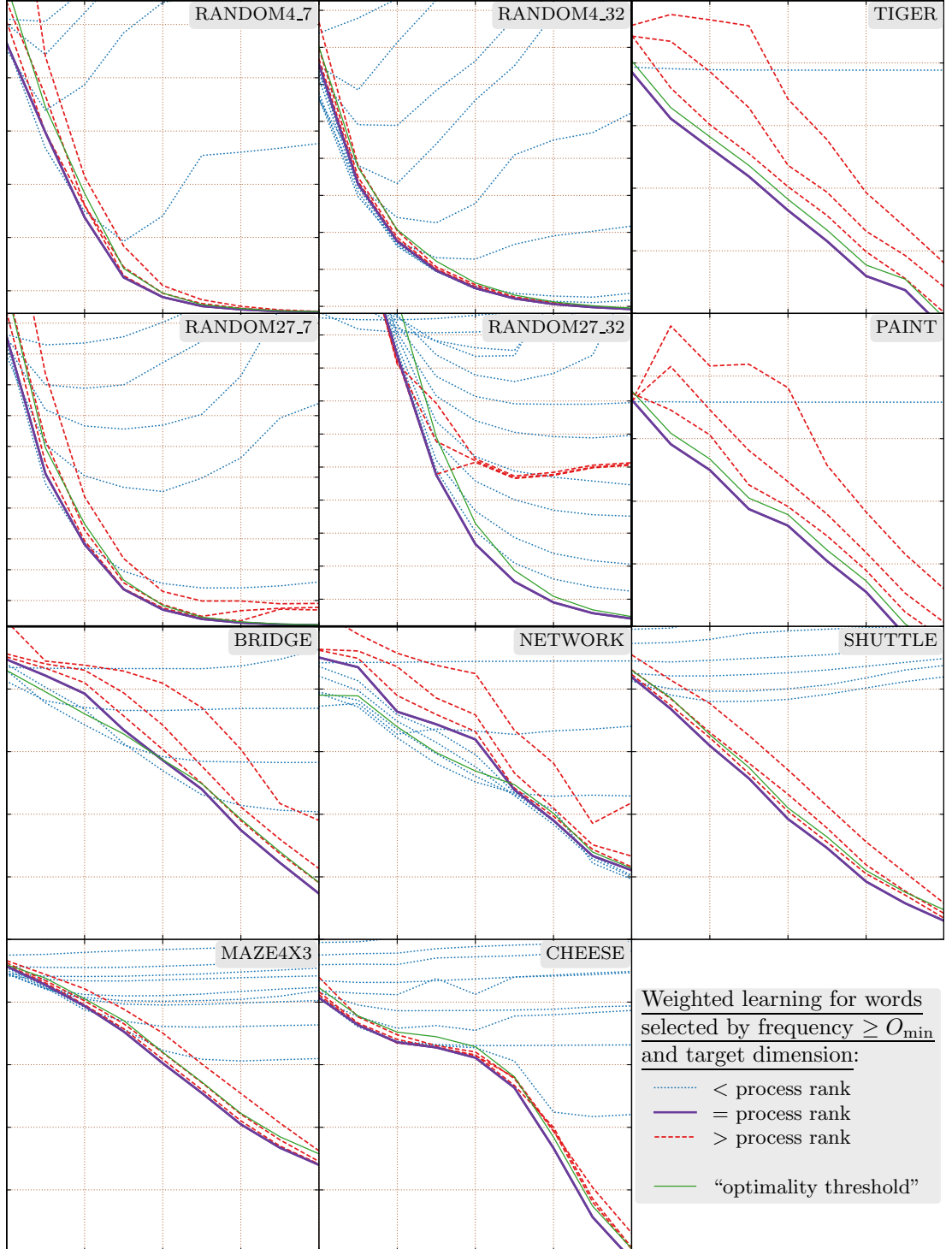


Figure 8: Evaluation of various target dimension settings using the weighted spectral algorithm with indicative and characteristic words selected by frequency greater or equal to $O_{\min}$ that is computed by the formulas presented in Section 5.2.1. For each training data length, the “optimality threshold” is the performance of the best model with an added tolerance of 0.25 times its relative entropy in the case of the stochastic process benchmarks, or else with an added tolerance of 0.5 times its average one-step prediction error.

6. EMPIRICAL INVESTIGATIONS

training data. We can then use these threshold curves to define *ranges* of suitable target dimensions that are more easily interpretable than merely the optimal target dimension.

The first observation we make is that using the process rank, which in these cases is finite and known, as target dimension in many cases gives the best or close to the best performance, especially for large amounts of training data, but often also when only limited training data is available.

Next, let us examine what happens when the target dimension is selected lower than the process rank, and let us consider Figure 7 first. We clearly see that the performance converges to some plateau value and remains at that value as the training data is increased, while larger target dimensions yield better plateau values for the performance. This is what we expect, and the likely interpretation is that reaching a plateau value for a target dimension d means that we have learned a best d -dimensional approximation to the process, i.e., the performance cannot improve further, since there simply is no better d -dimensional SS model for the process. Note that we cannot verify this (except for the 1-dimensional case), since we do not know how to compute the best low-dimensional SS approximation, for instance minimizing the cross-entropy, to a process given by a higher-dimensional SS representation. Furthermore, spectral learning, which is based on the learning Equations (2), is guaranteed to converge to a correct model if the process rank is used as the target dimension, but there is currently no theoretical guarantee that using a lower target dimension d will result in convergence to a best d -dimensional approximation. Interestingly, if we turn to the results for weighted spectral learning shown in Figure 8, we observe a different behavior. Here, for increasing training data lengths and a target dimension d less than the process rank, the performance first improves to a value close to the corresponding plateau value observed for standard spectral learning but then appears to generally decrease again, sometimes dramatically. We can observe this even when learning one-dimensional models. This is no cause for alarm, as this behavior is exhibited only when the choice of target dimension is clearly non-optimal anyway, i.e., in situations where some larger target dimension gives much better performance than the hypothetical plateau value. Also, we do still observe convergence to a correct model when using the process rank as target dimension. But this does highlight the importance of selecting an appropriate target dimension, and it is unclear what this means for learning real-world processes that may not have a finite rank at all. Finally, if we carefully inspect a particular case, namely RANDOM27_32 learned for a target dimension of 20 (six settings less than the process rank), we see that the performance appears to plateau at a value above 4.5 when using

standard spectral learning, but at some point reaches a performance better than 4.4 when using weighted spectral learning, thereby questioning the hypothesis that the plateau values correspond to best low-dimensional approximations.

Let us also comment on what happens if the target dimension is chosen larger than the process rank. As we might expect, the more we exceed the process rank, the worse the performance of the estimated model gets, but the effect on performance seems to be less than that of using a target dimension that is smaller than the process rank. For a given target dimension exceeding the process rank, the performance generally still improves as the amount of training data increases, and it appears that we usually retain convergence to a correct model. The exceptions are the BRIDGE benchmark in the case of the standard spectral algorithm that we consider here, where a too large target dimension may not lead to convergence, and, strikingly, the RANDOM27_32 benchmark when using the weighted spectral learning algorithm with indicative and characteristic words selected by frequency larger or equal to $O_{\min}$, in which case the performance using a target dimension just one greater than the process rank stagnates at a relatively poor performance for increasing training data lengths. A typical explanation for poor performance caused by a too large target dimension is overfitting, but that is not what is happening here, at least not in the usual sense. In the case of overfitting, we would expect the models estimated using larger target dimensions to perform much better when evaluated on the training data, but we have checked this (results not shown) and this is indeed not the case. Instead, we can monitor the model stability during evaluation by counting how often a setback operation is required and thereby observe that the models using a larger target dimension that perform poorly also exhibit poor stability. There may be numerical reasons for this, as the matrix $\hat{A}$ of coordinates for the columns of $\hat{F}$ projected to a lower-dimensional basis will be (nearly) rank deficient if the target dimension exceeds the process rank, especially when using more training data, which means the learning Equations (2) are essentially underdetermined. We have not covered this case, but a typical way to deal with this would be to employ some form of regularization when solving the learning equations. Instead, we will rely on the dimension estimation methods to avoid such cases altogether.

Conclusion 66. *The standard and weighted spectral learning algorithms with appropriate selections of indicative and characteristic words generally give good results when using the process rank as target dimension, which also guarantees convergence to a correct model, but limited training data may sometimes require using a lower target dimension to achieve optimal performance. However, the convergence properties when using a lower target dimension or a target dimension exceeding the process rank are*

6. EMPIRICAL INVESTIGATIONS

currently poorly understood. Using a target dimension that exceeds the process rank has no advantages but may cause stability issues and should generally be avoided.

We will now use the ranges of suitable target dimensions that we have established above to compare the methods for estimating a suitable target dimension. As before, we consider two concrete algorithms, namely in Figure 9 standard spectral learning with sets X, Y of indicative and characteristic words chosen by the words occurring in the training data of length at most L such that $|\Sigma|^L \leq 1024$, and in Figure 10 weighted spectral learning with indicative and characteristic words chosen as the at most 1024 words occurring at least $O_{\min}$ times in the training data, with $O_{\min}$ computed as previously explained. On the y -axis we show the target dimension instead of the performance evaluation, using a range from zero to five above the process rank, and we indicate the process rank as well as the ranges of suitable target dimensions. For each method under investigation, we can then plot the curve of its estimated target dimensions and see if it lies within the suitable range and approaches the process rank as the training data increases. The target dimension estimates we consider are the numerical ranks r_F , r_2 , r_{gap} , and r_J of the matrix $\hat{F}_{Y,X}$, each considered once with and once without the row and column scaling to normalize the element-wise variances as described in Section 5.3. Note that we perform this analysis for each of the three training sequences, so the ranges of suitable dimensions as well as the dimension estimates that are shown are averages. Also note that the curves are slightly offset from their actual value for better visualization.

Let us first comment on the numerical rank r_J that uses the relative error on matrix entries to compute a singular value cutoff (cf. Section 5.3), which is also the method originally proposed for the task. As we can see, this approach in many cases leads to gross over and under estimation of a suitable target dimension, even when large amounts of training data are available. While row and column scaling has a strong effect on the numerical rank r_J , the results remain unsatisfactory. We will therefore dismiss this approach.

Next, we consider the numerical rank r_{gap} that is based on identifying a gap in the singular value spectrum of $\hat{F}_{Y,X}$ starting from an initial dimension estimate, which to date is the only other published method for the task of estimating a suitable target dimension. This method will always give a dimension estimate greater than r_F , since we have used r_F as an initial dimension estimate. Unfortunately, this method often produces much too large target dimension estimates, apparently because it identifies and selects a gap only very far down in the singular value spectrum. This occurs with and without row and column scaling of $\hat{F}_{Y,X}$, and even in cases with sufficient

amount of training data (PAINT, BRIDGE and SHUTTLE in Figure 9 as well as CHEESE in both figures). A simple fix that we have attempted is to restrict this gap search to only the say 20 singular values below the r_F -th, which improves but does not resolve the issue. A more refined gap search method is therefore required, but it is also conceivable that in many cases, perhaps especially in the case of real-world data of infinite rank, there simply is no easily identifiable gap in the singular value spectrum corresponding to a suitable target dimension. Let us now consider the cases where this method does not overshoot and restrict ourselves to the version using row and column scaling, which appears to give slightly better dimension estimates. In these cases, r_{gap} performs similarly to r_2 , sometimes better, as in the RANDOM27.32 benchmark, and sometimes worse, as in the NETWORK benchmark in Figure 9 or the CHEESE benchmark in Figure 10. Overall, the additional gap search employed for r_{gap} does not appear to offer significant advantages over simply using the numerical rank r_2 but suffers from an overshoot issue that currently renders it too unreliable.

Finally, we turn to the numerical ranks r_F and r_2 , which we consider together, since they share many properties. First of all, we can see that the dimension estimates by r_F and r_2 are significantly improved by the row and column scaling of $\hat{F}_{Y,X}$ that we have proposed to normalize the variances of the matrix entries. We will therefore only further consider the estimates using this normalization. Next, both r_F and r_2 generally do not overshoot a suitable target dimension, and if they do, then by at most one dimension. However, they may underestimate a suitable target dimension, with r_F usually being lower than r_2 . The numerical ranks r_F and r_2 therefore provide conservative estimates for a suitable dimension, as we had expected. Because of this, we may combine both estimates and use instead $\max\{r_F, r_2\}$ to estimate a suitable target dimension. Unfortunately, this still underestimates a suitable target dimension in several cases, but only slightly. Importantly, as the training data increases, $\max\{r_F, r_2\}$ indeed converges to a suitable target dimension (or just below in the RANDOM4.32 benchmark in Figure 10). While this is clearly not optimal, it does appear to be a reasonable and importantly reliable method to estimate a suitable target dimension.

Conclusion 67. *The previously known methods to estimate a target dimension using the numerical ranks r_J as well as r_{gap} give unsatisfactory results on the benchmark problems considered here. In contrast, $\max\{r_F, r_2\}$, which uses the proposed numerical ranks r_F and r_2 , when combined with the novel variance normalization by row and column scaling of $\hat{F}_{Y,X}$ gives a conservative estimate for a target dimension that is reasonably close to an optimal target dimension. Importantly, this converges to a suitable target dimension as the amount of training data is increased.*

6. EMPIRICAL INVESTIGATIONS

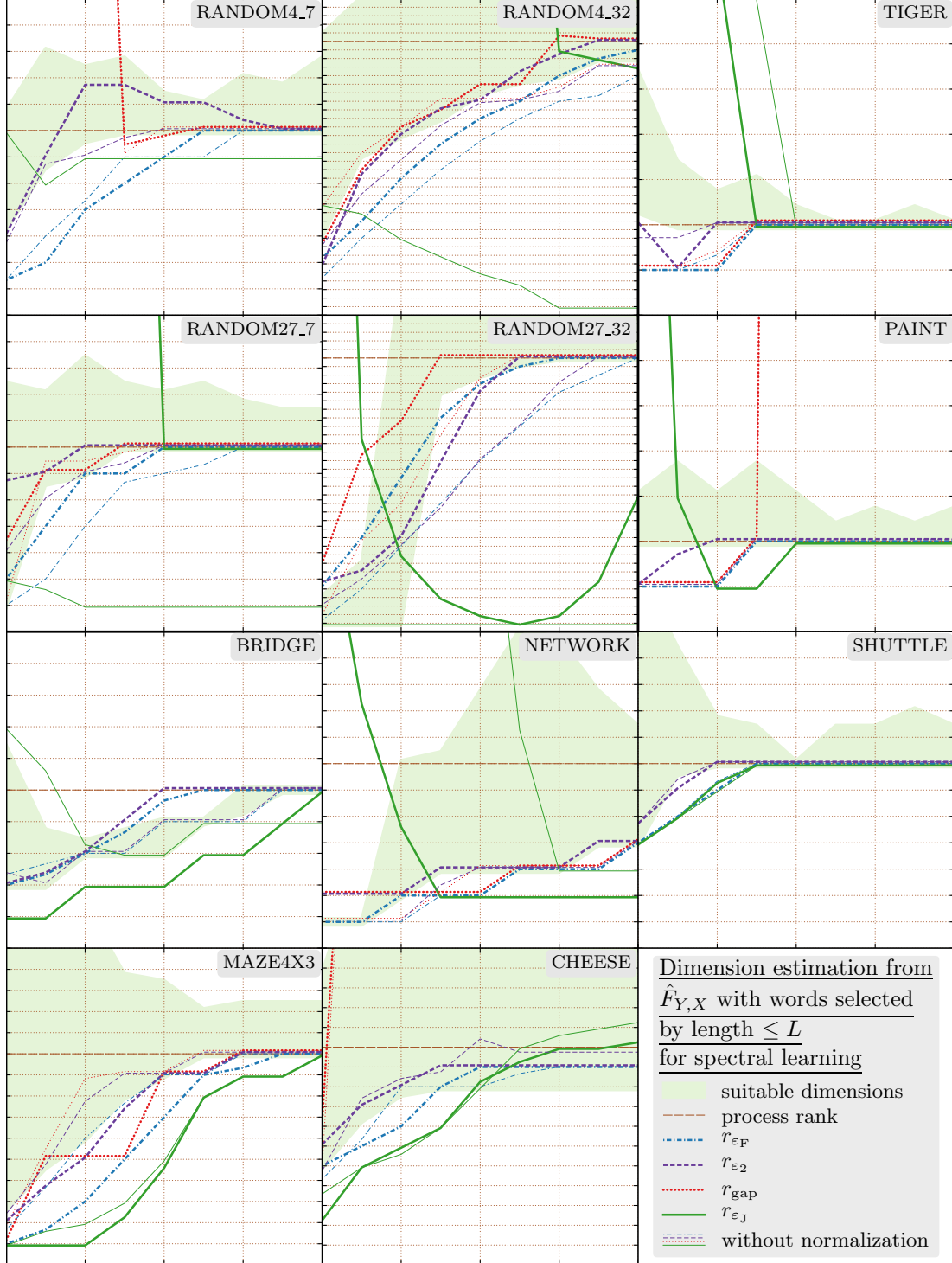


Figure 9: Evaluation of target dimension estimates by the numerical ranks r_F , r_2 , r_{gap} , and r_J of the matrix $\hat{F}_{Y,X}$ either with or without variance normalization by row and column scaling. The indicative and characteristic words X, Y are selected as the words occurring in the training data of length less than or equal to L for the largest L such that $|\Sigma|^L \leq 1024$. The suitable dimensions are derived using the “optimality threshold” and the results for spectral learning with various target dimensions from Figure 7.

6.3 Selecting the Target Dimension

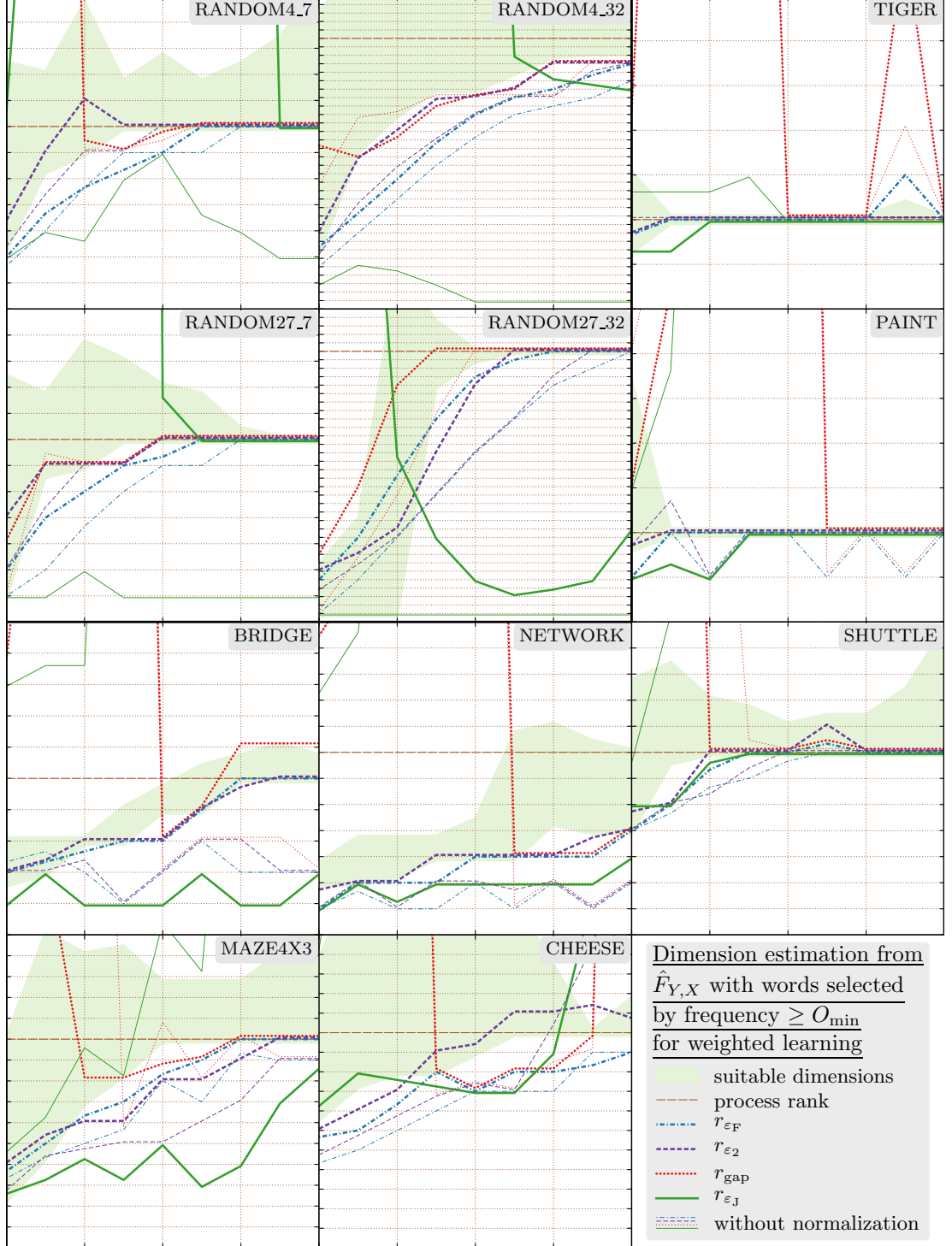


Figure 10: Evaluation of target dimension estimates by the numerical ranks r_F , r_2 , r_{gap} , and r_J of the matrix $\hat{F}_{Y,X}$ either with or without variance normalization by row and column scaling. The indicative and characteristic words X, Y are the at most 1024 words occurring at least O_{min} times in the training data, with O_{min} computed by the formulas presented in Section 5.2.1. The suitable dimensions are derived using the “optimality threshold” and the results for weighted learning with various target dimensions from Figure 8.

6. EMPIRICAL INVESTIGATIONS

6.4 Comparison of Weighted Spectral Learning Variants

We will now compare the performance of the versions of weighted spectral learning that we have described in Section 5.4. The algorithms that we consider are the standard spectral learning Algorithm 6 (which is equivalent to the EC Algorithm 10), the row and column weighted spectral learning Algorithm 9, the (generalized) ES Algorithm 11, and the weighted spectral learning Algorithm 8, including the simplified element-wise weighted spectral learning variant. For the ES algorithm, we use the model estimated by the standard spectral algorithm as initial model estimate and perform five iterations of the ES Algorithm 11, from which we then select the model that gives the highest likelihood measured on the training data, as suggested in Section 5.4.6.

In the previous Sections 6.2 and 6.3 we focussed on investigating methods to select indicative and characteristic words and to estimate a suitable target dimension, but especially in Section 6.2 we have already seen a direct comparison between standard and weighted spectral learning. We concluded that weighted spectral learning promises to be especially useful and leads to the overall best results when combined with a selection of indicative and characteristic words by the at most $Z_{\max} = 1024$ words occurring most often in the training data with an occurrence frequency that is greater or equal to $O_{\min}$, where $O_{\min}$ is computed according to the heuristic presented in Section 5.2.1. We will therefore use this method of selecting sets X, Y of indicative and characteristic words for our current comparison and combine it with a target dimension estimated by the numerical rank $\max\{r_F, r_2\}$ computed from $\hat{F}_{Y,X}$ after variance normalization using row and column scaling, which has produced the most suitable dimension estimates in the previous Section 6.3. These approaches to selecting indicative and characteristic words as well as the target dimension require no prior knowledge about the process and no tweaking of parameters and have proven to be generally applicable on all synthetic benchmarks. Using these approaches, we can therefore now also include the real-word benchmarks into our comparison.

The evaluations on the synthetic benchmarks are shown in Figure 11, and the results on the real-world benchmarks are shown in Figure 12, using the format that we have explained in Section 6.1.3.

As we have already observed in Section 6.2 for a target dimension fixed to the process rank, using weighted spectral learning leads to an improved performance over standard spectral learning for all synthetic benchmarks, while here the improvement is particularly striking on the benchmarks for controlled processes. The latter, however, is mainly due to the way we have selected indicative and characteristic words. The results

6.4 Comparison of Weighted Spectral Learning Variants

from Section 6.2 suggest that we should only expect a slight but consistent improvement in performance in the case of controlled processes by using weighted learning if we were to use a more traditional method of selecting indicative and characteristic words. Nevertheless, as we have discussed, this is not always feasible (e.g., CHEESE benchmark), and the fact that weighted learning works well with the generally applicable method of selecting indicative and characteristic words that we employ here is a major advantage over the standard spectral learning algorithm in itself. On the real-world benchmarks, which are benchmarks for stochastic processes, weighted learning leads to improved performance on the ECOLI benchmark, but, in fact, to worse performance on the BIBLE27 benchmark.

Next, we can observe that the row and column weighted spectral algorithm in most cases gives almost identical performance to the ES algorithm, which is not surprising, as we have shown in Section 5.4.6 that they are closely related. The ES algorithm does perform slightly better than the row and column weighted spectral algorithm and on par with the weighted spectral algorithm on the real-world benchmarks ECOLI and BIBLE27 in the case of limited training data.

Similarly, we can see that the simpler element-wise weighted spectral algorithm in many cases gives identical performance to the weighted spectral algorithm, especially in the case of limited training data. Only in some benchmarks (BRIDGE, SHUTTLE, CHEESE) and for larger amounts of training data we observe a better performance by the weighted spectral algorithm. We conjecture that the covariance information on the coordinate representations $\hat{A}_z$ of the matrices $\hat{F}_{zY,X}$ that is used by the weighted spectral algorithm but ignored by the element-wise weighted spectral algorithm can only be estimated in a meaningful way for sufficiently large amounts of training data.

On the BIBLE27 benchmark for large amounts of training data ($\geq 10^{5.5}$ symbols), we could not perform weighted spectral learning with the estimated target dimension (204, 279 and 372, respectively) due to the computational complexity of solving the GLS subproblems in Step (iii) of the weighted spectral Algorithm 8. Instead, we performed the weighted spectral learning using the largest target dimension we could reasonably handle, which was 192. This has a striking effect. The performance only slightly improves with training data of length 10^6 compared to $10^{5.5}$, and quite drastically degrades when using the complete available training data of length $3.8 \cdot 10^6$. This resembles the effect we have observed and commented on in Section 6.3 when performing weighted spectral learning with a too small target dimension for the available data. It may be possible to overcome this problem by employing a more efficient GLS solver or by using regularization when solving the GLS sub-problems, but this requires further

6. EMPIRICAL INVESTIGATIONS

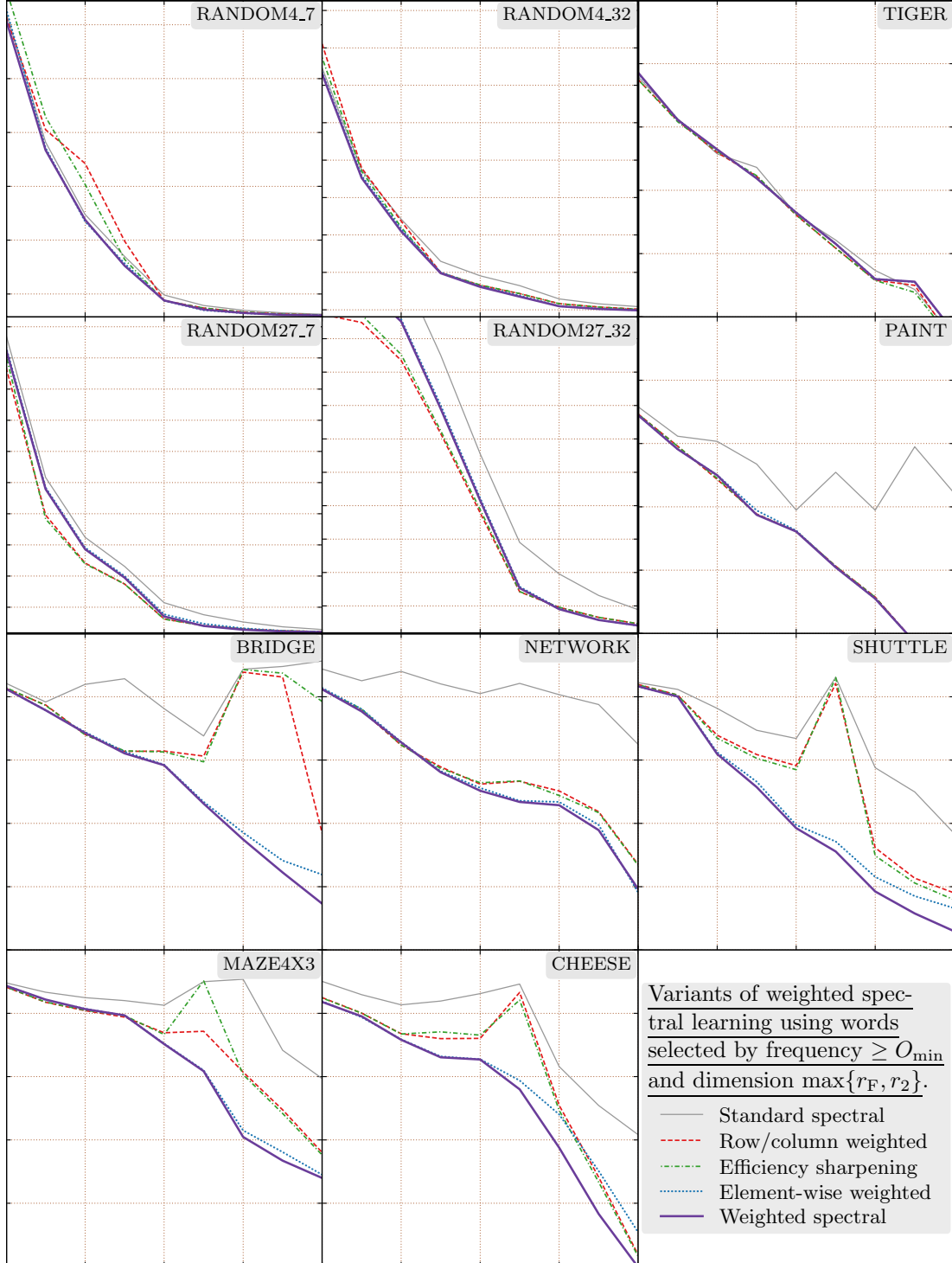


Figure 11: Evaluation of weighted spectral learning variants on the synthetic benchmarks using the most promising methods for selecting indicative and characteristic words as well as a suitable target dimension as established in the previous Sections 6.2 and 6.3.

6.4 Comparison of Weighted Spectral Learning Variants

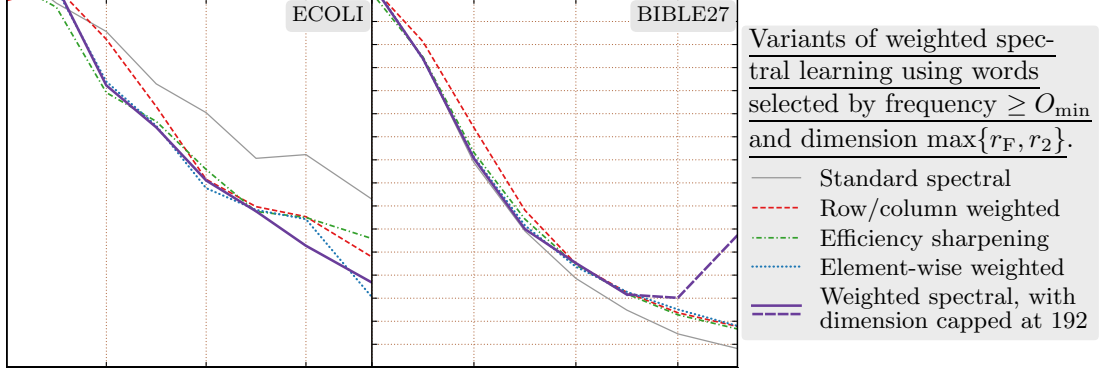


Figure 12: Evaluation weighted spectral learning variants on the real-word benchmarks using the most promising methods for selecting indicative and characteristic words as well as a suitable target dimension as established in the previous Sections 6.2 and 6.3. Note that we cap the target dimension at 192 in the case of weighted spectral learning due to issues of computational complexity, which is relevant for training sequences of length $\geq 10^{5.5}$ in the case of the BIBLE27 benchmark, where the estimated target dimension exceeds 192.

investigation. Currently, we must recommend using weighted spectral learning only with a target dimension setting that is appropriate for the available training data, and if this is not possible then to resort to a different algorithm, for instance element-wise weighted spectral learning, which often gives comparable performance.

Finally, we compare the performance of the row and column weighted spectral algorithms (including the ES variant) to the weighted spectral algorithm. On the benchmarks for controlled processes, the performance of the row and column weighted spectral algorithms is somewhere between the performance of the standard spectral algorithm and the weighted spectral algorithm. In all of these cases (except TIGER), the performance is clearly better than that of the standard spectral algorithm, and in some cases it is equal to (PAINT) or approaches (NETWORK) the performance of the weighted algorithms. On the TIGER benchmark all algorithms lead to the same performance. On the benchmarks for stochastic processes, the performance of the row and column weighted spectral algorithms appears to be broadly comparable to that of the weighted spectral algorithm, but there are exceptions. On the RANDOM27_7 benchmark the row and column weighted algorithms outperform the weighted spectral algorithm in the case of limited training data, while on the RANDOM4_7 benchmark with limited training data they are clearly worse, performing even worse than the standard spectral algorithm. And on the ECOLI benchmark when considering longer training data, the row and column weighted algorithms cannot quite reach the performance

6. EMPIRICAL INVESTIGATIONS

of the weighted spectral algorithms but still clearly outperform the standard spectral algorithm, similar to what we have observed for the controlled process benchmarks.

Conclusion 68. *The row and column weighted and ES algorithm have similar performance with a slight advantage for the ES algorithm in some cases. The element-wise weighted and weighted spectral algorithm have similar performance, with an advantage for the weighted spectral algorithm in the case of controlled processes and large training data. However, weighted spectral learning currently cannot be used for very large target dimensions (≥ 192).*

Any type of weighted spectral learning typically shows an improvement over standard spectral learning, in particular enabling the use of more general selection strategies for indicative and characteristic words, while the weighted spectral learning algorithm outperforms the row and column weighted spectral algorithm on the benchmarks for controlled processes and shows similar performance otherwise.

However, the comparative performance of the various algorithms depends on the particular process being modeled, with the row and column weighted spectral algorithm (as well as the ES algorithm) giving the overall best performance on the RANDOM27_7 benchmark and the standard spectral algorithm clearly outperforming all weighted algorithms on the BIBLE27 benchmark.

Learning with Missing Values

In this chapter we will examine how to use (weighted) spectral learning algorithms to estimate OOMs for stochastic processes from training data with missing values. Dealing with missing values is straight-forward when learning HMMs via EM (Baum-Welch algorithm), but this has so far not been addressed for spectral learning of OOMs¹. We fill this gap, which might help boost the practical relevance of spectral learning algorithms in the context of stochastic processes.

More precisely, let us assume we draw an initial sample $\bar{x} = x_1 \dots x_N$ from a stochastic process with alphabet Σ that is described by a d -dimensional OOM $\mathcal{M}$. However, for some time steps t we do not observe the value x_t , which we call a missing value. Let $\bar{m} := m_1 \dots m_N$ be a sequence of “missingness indicators”, with $m_t = 0$ if the value x_t at time t is observed, and $m_t = 1$ if it is missing. Let $\bar{x}^{\text{obs}}$ denote the observed values of the sample $\bar{x}$, and let $\bar{x}^{\text{mis}}$ denote the missing values. Our goal is then to adapt the spectral learning algorithms to estimate a model for the underlying stochastic process using $\bar{m}$ and $\bar{x}^{\text{obs}}$ as training data.

Depending on how the missingness $\bar{m}$ depends on $\bar{x}$, one typically distinguishes the following three cases:

- Missing completely at random (MCAR): The missingness probability is independent of the observed and missing data, i.e., $\mathbb{P}(\bar{m} | \bar{x}) = \mathbb{P}(\bar{m})$.
- Missing at random (MAR): The missingness probability depends only on the observed data, i.e., $\mathbb{P}(\bar{m} | \bar{x}^{\text{obs}}, \bar{x}^{\text{mis}}) = \mathbb{P}(\bar{m} | \bar{x}^{\text{obs}})$ for all $\bar{x}^{\text{mis}}$.

¹ Balle and Mohri [2012] use constrained matrix completion to address spectral learning with missing entries in the system matrix $\hat{F}_{Y,X}$, but that is a different problem. In fact, that problem can also be addressed with weighted learning by assigning zero weight to the missing entries.

7. LEARNING WITH MISSING VALUES

- Missing not at random (MNAR): The missingness probability can depend on the the missing data.

These definitions go back to Rubin [1976], who showed that MAR is a key requirement to be able to “ignore” the missingness mechanism, in the sense that a likelihood-based estimate of the parameters θ for the distribution underlying $\mathbb{P}(\bar{x} | \theta)$ based only on the likelihood $\mathbb{P}(\bar{x}^{\text{obs}} | \theta)$ will give the same result as an estimate based on the joint likelihood $\mathbb{P}(\bar{m}, \bar{x}^{\text{obs}} | \theta, \phi)$, where ϕ are the parameters of the missingness distribution underlying $\mathbb{P}(\bar{m} | \bar{x}, \phi)$. For this, Rubin [1976] required MAR not only for the true missingness distribution but for all potential missingness distributions parameterized by ϕ . A further requirement is that the parameters θ and ϕ must be “distinct”. Interestingly, in the context of likelihood-based parameter estimation, it is sufficient to require MAR only as a property of the data sample $\bar{m}, \bar{x}^{\text{obs}}$ and not for the underlying missingness distribution. Unfortunately, with regard to these mentioned aspects, the term “missing at random” is used inconsistently throughout the literature, and it is often not clear what exactly is meant or required. A mathematically careful account of the various versions of MAR and the intricacies involved is given by Seaman et al. [2013].

In the context of spectral learning of OOM parameters, where we are dealing with sequential data, we will require the missingness mechanism to be *always missing sequentially* at random (AMSAR), which we define as follows:

Definition 69. *The missingness is always missing sequentially at random (AMSAR) if for all $t, \bar{m}, \bar{x}$ we have¹*

$$\mathbb{P}(m_t | \bar{m}_{t-1}, \bar{x}) = \mathbb{P}(m_t | \bar{m}_{t-1}, \bar{x}_{t-1}^{\text{obs}}), \quad \text{where}$$

$\bar{m}_{t-1} = m_1 \dots m_{t-1}$, and $\bar{x}_{t-1}^{\text{obs}}$ denotes the observed values among the first $t-1$.

The missingness is strictly uncertain if $\mathbb{P}(m_t | \bar{m}_{t-1}, \bar{x}) > 0$ for all $t, \bar{m}, \bar{x}$.

The AMSAR requirement says that the missingness at every time t is conditionally independent of the current and future values of the process (observed or not), conditioned on the past missingness and the past observed values. This means that the missingness at time t does not provide any additional information about the future of the process that is not already known from the past observations. Therefore:

Lemma 70. *AMSAR missingness implies that for any $t, \bar{m}, \bar{x}$, we have*

$$\mathbb{P}(x_t | \bar{m}_t, \bar{x}_{t-1}^{\text{obs}}) = \mathbb{P}(x_t | \bar{x}_{t-1}^{\text{obs}}). \quad (17)$$

¹We will interpret statements about conditional probabilities to mean that they must hold only for the cases that the conditional probabilities are well-defined.

Proof. We assume that $\mathbb{P}(\bar{m}_t, \bar{x}_{t-1}^{\text{obs}}, x_t) > 0$, as otherwise the statement is trivial. Then $\mathbb{P}(x_t | \bar{m}_t, \bar{x}_{t-1}^{\text{obs}}) = \mathbb{P}(\bar{m}_t | x_t, \bar{x}_{t-1}^{\text{obs}}) \mathbb{P}(x_t | \bar{x}_{t-1}^{\text{obs}}) / \mathbb{P}(\bar{m}_t | \bar{x}_{t-1}^{\text{obs}}) = \mathbb{P}(x_t | \bar{x}_{t-1}^{\text{obs}})$, because $\mathbb{P}(\bar{m}_t | x_t, \bar{x}_{t-1}^{\text{obs}}) = \prod_{k=1}^t \mathbb{P}(m_k | \bar{m}_{k-1}, x_t, \bar{x}_{t-1}^{\text{obs}}) \stackrel{(*)}{=} \prod_{k=1}^t \mathbb{P}(m_k | \bar{m}_{k-1}, \bar{x}_{t-1}^{\text{obs}}) = \mathbb{P}(\bar{m}_t | \bar{x}_{t-1}^{\text{obs}})$, where for $(*)$ we have used that the AMSAR condition $\mathbb{P}(m_k | \bar{m}_{k-1}, \bar{x}) = \mathbb{P}(m_k | \bar{m}_{k-1}, \bar{x}_{k-1}^{\text{obs}})$ also implies that $\mathbb{P}(m_k | \bar{m}_{k-1}, x_t, \bar{x}_{t-1}^{\text{obs}}) = \mathbb{P}(m_k | \bar{m}_{k-1}, \bar{x}_{t-1}^{\text{obs}})$. ■

Let us now rewrite the data $\bar{m}, \bar{x}^{\text{obs}}$ as an input-output sequence $\bar{y} = m_1 o_1 \dots m_N o_N$ over the alphabet $\{0, 1\} \times (\Sigma \cup \{\emptyset\})$ by setting $o_t = x_t$ if $m_t = 0$, i.e., if the observation at time t is not missing, and $o_t = \emptyset$ otherwise. We will call any sequence $m_1 o_1 \dots m_K o_K$ over the alphabet $\{0, 1\} \times (\Sigma \cup \{\emptyset\})$ *consistent* if it satisfies $o_t = \emptyset$ whenever $m_t = 1$ and $o_t \neq \emptyset$ whenever $m_t = 0$. We will interpret the observation process with missing data as an input-output process, where we view the missingness as input and the observation (possibly $\emptyset$) as output. We can then factorize the distribution of the combined input-output process as in Equation (3) as

$$\mathbb{P}(\underbrace{m_1 o_1 \dots m_N o_N}_{\bar{y}}) = \underbrace{\prod_{t=1}^N \mathbb{P}(m_t | m_1 o_1 \dots m_{t-1} o_{t-1})}_{\pi(\bar{y})} \underbrace{\prod_{t=1}^N \mathbb{P}(o_t | m_1 o_1 \dots m_{t-1} o_{t-1} m_t)}_{f(\bar{y})}.$$

Note that $\mathbb{P}(\bar{m}, \bar{x}^{\text{obs}}) = \mathbb{P}(m_1 o_1 \dots m_N o_N)$ for all $\bar{m}, \bar{x}^{\text{obs}}$, which implies that $\mathbb{P}(\bar{y}) = 0$ if $\bar{y}$ is inconsistent. Assuming AMSAR missingness, for all consistent $\bar{y}$ the input policy $\pi(\bar{y}) = \prod_{t=1}^N \mathbb{P}(m_t | \bar{m}_{t-1}, \bar{x}_{t-1}^{\text{obs}}) = \mathbb{P}(\bar{m} | \bar{x})$ describes the missingness mechanism, while according to Lemma 70 and noting that for consistent sequences we have

$$\mathbb{P}(o_t | m_1 o_1 \dots m_{t-1} o_{t-1} m_t) = \begin{cases} \mathbb{P}(x_t | \bar{m}_t, \bar{x}_{t-1}^{\text{obs}}) & \text{if } m_t = 0, \\ 1 & \text{if } m_t = 1, \end{cases} \quad (18)$$

we see that the controlled process f satisfies

$$\forall \bar{y} : f(\bar{y}) = \begin{cases} \mathbb{P}(\bar{x}^{\text{obs}}) & \text{if } \bar{y} \text{ is consistent,} \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

Note, however, that $f(\bar{y})$ is well-defined only if $\pi(\bar{y}) \neq 0$, which by our interpretation of conditional probabilities means that the above Equation (19) only states equality for the cases where $\pi(\bar{y}) \neq 0$. We will therefore simply require $\pi > 0$, as we have done for the estimators from Remark 37 that are used in the IO-OOM learning algorithms,

7. LEARNING WITH MISSING VALUES

i.e., we require the missingness to be strictly uncertain. It is important to note that $\pi > 0$ is a sufficient but not a necessary requirement, and we will later consider a case empirically where the missingness is not strictly uncertain.

As a special case of Equation (19) we have $f(0x_1 \dots 0x_N) = \mathbb{P}(\bar{x})$ for all $\bar{x} \in \Sigma^*$. This warrants the following Algorithm 13 to estimate a model $\hat{\mathcal{M}}$ from data with AMSAR and strictly uncertainly missing values, which learns an IO-OOM for the controlled process f from the data with missing values viewed as an input-output sequence, and reduces this to an OOM by only considering inputs $m_t = 0$, i.e., non-missing values.

Algorithm 13: Learning with AMSAR and strictly uncertainly missing values

In : An initial sample $\bar{m}, \bar{x}^{\text{obs}}$ from a stochastic process with AMSAR and strictly uncertainly missing values

Out: A model $\hat{\mathcal{M}}$ for the stochastic process

- 1 Write the data as an input-output sequence $\bar{s} = m_1 o_1 \dots_N o_N$ with

$$o_t = \begin{cases} x_t & \text{if } m_t = 0, \\ \emptyset & \text{if } m_t = 1. \end{cases}$$

- 2 Learn an IO-OOM $(\hat{\sigma}, \{\hat{\tau}_{m,o}\}, \hat{\omega}_\varepsilon)$ over the alphabet $\{0, 1\} \times (\Sigma \cup \{\emptyset\})$ from $\bar{s}$.
 - 3 Return the model $\hat{\mathcal{M}} = (\hat{\sigma}, \{\hat{\tau}_x\}, \hat{\omega}_\varepsilon)$ over the alphabet Σ , where $\hat{\tau}_x = \hat{\tau}_{0,x}$.
-

If we recall that all of the spectral learning algorithms estimate the matrices $\hat{\tau}_{m,o}$ only in their last step by solving a set of linear equations separately for each $\hat{\tau}_{m,o}$, we realize that in Algorithm 13 we do not need to compute the estimates $\hat{\tau}_{1,x}$ for $x \in \Sigma$ and $\hat{\tau}_{0,\emptyset}$ at all. In other words, the Steps 2 and 3 of Algorithm 13 taken together can be seen as a spectral learning algorithm that returns the desired OOM estimate directly. However, the indicative and characteristic words that we need to consider will be over the alphabet $\{0, 1\} \times (\Sigma \cup \{\emptyset\})$, and importantly the input-output interpretation allows us to obtain the required estimates $\hat{f}(\bar{y})$ for such words $\bar{y}$ via Remark 37.

Next, we note that in the case of AMSAR and strictly uncertainly missing values, the controlled process f is fully determined by the stochastic process governing $\bar{x}$ via Equation (19). In fact, if $\mathcal{M} = (\sigma, \{\tau_x\}, \omega_\varepsilon)$ is a d -dimensional model for the stochastic process governing $\bar{x}$, then we can easily construct a model $\mathcal{M}' = (\sigma', \{\tau'_{m,o}\}, \omega'_\varepsilon)$ for the controlled process f by setting

$$\begin{aligned} \sigma' &= \sigma, \quad \forall x \in \Sigma : \tau'_{0,x} = \tau_x, \quad \tau'_{1,\emptyset} = \sum_{x \in \Sigma} \tau_x, \\ \omega'_\varepsilon &= \omega_\varepsilon, \quad \forall x \in \Sigma : \tau'_{1,x} = 0, \quad \tau'_{0,\emptyset} = 0. \end{aligned}$$

It is straight forward to check that $f_{\mathcal{M}'}$ is a controlled process and that $f_{\mathcal{M}'}(\bar{y}) = \sum_{\bar{x}^{\text{mis}}} \mathbb{P}(\bar{x}^{\text{obs}}, \bar{x}^{\text{mis}}) = \mathbb{P}(\bar{x}^{\text{obs}})$ for all consistent $\bar{y}$, as well as $f_{\mathcal{M}'}(\bar{y}) = 0$ for inconsistent $\bar{y}$. This shows that the rank of the controlled process f is the same as the rank of the stochastic process governing $\bar{x}$. Taken together with the above comment that only the matrices $\tau_{0,x}$ for $x \in \Sigma$ need to be estimated, we see that this approach of estimating a model $\hat{\mathcal{M}}$ via modeling the controlled process does not increase the model complexity.

7.1 Current Restrictions

Recall that for learning IO-OOMs we have required the combined input-output process to be stationary and ergodic, as well as the input policy π to satisfy $\pi > 0$. None of these requirements is strictly necessary, but the more general cases have not been studied. One practically relevant set of conditions that satisfies these requirements is the following:

- The data $\bar{x}$ are an initial sample from a stationary and ergodic stochastic process $\bar{X}$ that can be described by a d -dimensional OOM.
- The missingness indicators $\bar{m}$ are an initial sample from a stationary and ergodic stochastic process $\bar{M}$ that is independent of $\bar{X}$.
- At least one of the processes $\bar{X}$ or $\bar{M}$ is weakly mixing.
- For all $\bar{m}$, we have $\mathbb{P}(\bar{m}) > 0$.

The first three conditions ensure that the joint process $(\bar{X}, \bar{M})$ is stationary and ergodic, and therefore the combined input-output process governing the training data $\bar{s} = (\bar{m}, \bar{x}^{\text{obs}})$, which is a function of the joint process $(\bar{X}, \bar{M})$, is stationary and ergodic as well. The second condition ensures that the missingness is AMSAR (in fact, it will be always missing *completely* at random), and the last condition ensures that the missingness is strictly uncertain, i.e., that $\pi > 0$. At present, we will require these conditions when learning an OOM model for a stochastic process from data with missing values via Algorithm 13. Most importantly, this means we can currently use this approach only in the case of MCAR missingness. However, we should emphasize that this restriction is not due to the approach used by Algorithm 13 but due to the fact that IO-OOM learning in the case of non-stationary data has not been worked out.

7. LEARNING WITH MISSING VALUES

7.2 Modified Estimates $\hat{f}(\bar{y})$

When regarding the training data with missing values as an input-output sequence, we should slightly tweak the estimators from Remark 37 for the quantities $f(\bar{y})$ to take into account the fact that the symbols $(0, \emptyset)$ and $(1, x)$ for $x \in \Sigma$ can never occur. Furthermore, we can make use of Equation (17) that holds in the case of AMSAR missingness to modify and possibly improve the estimators from Remark 37.

Recall that by Remark 37 we estimate $f(\bar{y})$ for $\bar{y} = m_1 o_1 \dots m_K o_K$ by

$$\hat{f}(\bar{y}) = \prod_{k=1}^{|\bar{y}|} \hat{p}_k(\bar{y}) \quad \text{with } \hat{p}_k(\bar{y}) = \begin{cases} \frac{\#(m_1 o_1 \dots m_k o_k)}{\#(m_1 o_1 \dots m_k)} & \text{if } \#(m_1 o_1 \dots m_k) > 0, \\ \frac{1}{|\Sigma_O|} & \text{otherwise,} \end{cases}$$

where $\hat{p}_k(\bar{y})$ are estimates for $p_k(m_1 o_1 \dots m_k o_k) := \mathbb{P}(o_k \mid m_1 o_1 \dots m_{k-1} o_{k-1} m_k)$.

We first modify the estimates of $p_k(\bar{y})$ as follows to reflect the fact that the symbols $(0, \emptyset)$ and $(1, x)$ for $x \in \Sigma$ cannot occur:

$$\hat{p}_k(\bar{y}) = \begin{cases} 1 & \text{if } m_k = 1 \text{ and } o_k = \emptyset, \\ \frac{\#(m_1 o_1 \dots m_k o_k)}{\#(m_1 o_1 \dots m_k)} & \text{if } \#(m_1 o_1 \dots m_k) > 0 \text{ and not } (m_k = 1 \text{ and } o_k = \emptyset), \\ \frac{1}{|\Sigma|} & \text{if } \#(m_1 o_1 \dots m_k) = 0 \text{ and } m_k = 0 \text{ and } o_k \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \quad (20)$$

Now, in the case of AMSAR missingness, Lemma 70 implies that $\mathbb{P}(x_k \mid \bar{m}_k, \bar{x}_{k-1}^{\text{obs}}) = \mathbb{P}(x_k \mid \bar{m}_{k-1}^{\text{obs}}, \bar{x}_{k-1}^{\text{obs}}, m_k)$, where $\bar{m}_{k-1}^{\text{obs}}$ denotes the set of missingness indicators that are equal to 0 among $\bar{m}_{k-1}$. Taken together with Equation (18), this allows us to obtain estimates for $p_k(\bar{y})$ by conditioning on all occurrences of the *pattern* $m_{k-1}^{\text{obs}} \bar{x}_{k-1}^{\text{obs}} m_k$ in the training sequence $\bar{s}$. That is, for the missing values in $\bar{y}$, we can allow any symbol to occur, not just $(1, \emptyset)$, i.e., we may treat missing values as wildcards. Formally, we will replace the occurrence counts $\#(\cdot)$ in Equation (20) by the modified occurrence counts $\#^{\text{obs}}(\cdot)$ given by

$$\#^{\text{obs}}(\bar{y}) = \sum_{\bar{z} \stackrel{\text{obs}}{\succ} \bar{y}} \#(\bar{z}), \quad (21)$$

where we sum over all words $\bar{z}$ of the same length as $\bar{y} = m_1 o_1 \dots m_k o_k$ that match the observed values of $\bar{y}$, denoted by $\bar{z} \stackrel{\text{obs}}{\succ} \bar{y}$, which means that $z_t = y_t$ whenever $m_t = 0$. Clearly, the resulting estimators have the same requirement that the combined input-output process must be stationary and ergodic and that the missingness is strictly

uncertain as the estimators from Remark 37. The basic motivation for the use of these estimators is that they are based on more occurrences in the training data and should, therefore, typically have a higher precision.

We will use the same variance estimates for these estimators as derived in Section 5.1.1, replacing the counts $\#(\cdot)$ by the modified counts $\#^{\text{obs}}(\cdot)$ throughout if treating missing values as wildcards, with the exception that where we know the value of $p_k(\bar{y})$ exactly, i.e., when $m_k = 1$, we use this exact value for $\hat{p}_k(\bar{y})$ and a corresponding variance estimate of zero.

Computing the modified counts $\#^{\text{obs}}(\bar{y})$ naively via Equation (21) incurs a computational cost that is exponential in the number of missing values in $\bar{y}$. We will, however, attempt to avoid selecting words with many missing values as indicative and characteristic words, so that this computation remains tractable.

7.3 Empirical Investigation

We will now evaluate the performance of Algorithm 13 empirically by employing it to learn OOM models for stochastic processes from initial sample data with missing values. For this, we will use the stochastic process benchmarks from Section 6.1.2 but add missing values to the data as follows.

We use two different missingness mechanisms, and for each we consider missing value rates r_{mis} of $\frac{1}{9.5}$, $\frac{1}{4.5}$, and $\frac{1}{2.5}$. For the first missingness mechanism that we denote *randomly missing values*, we use an iid missingness process with missing probability according to the desired missing value rate r_{mis} . For the second missingness mechanism that we denote by *regularly missing values*, we use a missingness process where a value is missing every $\frac{1}{r_{\text{mis}}}$ time steps, i.e., sometimes the number of time steps between missing values is $\lfloor \frac{1}{r_{\text{mis}}} \rfloor$, and sometimes $\lceil \frac{1}{r_{\text{mis}}} \rceil$, such that the missing value rate is r_{mis} . This is achieved by a hidden Markov model governing the missingness process with $n = \lceil \frac{1}{r_{\text{mis}}} \rceil$ states, where state s_1 emits 1 (missing), while all other states emit 0 (not missing), and with transition structure as shown in Figure 13. The transition probability from state s_{n-1} to s_n is set to $p = \frac{1}{r_{\text{mis}}} - n + 1$. We can easily check that the expected number of time steps between the occurrence of ‘1’s is $(1-p)(n-1) + pn = \frac{1}{r_{\text{mis}}}$, so the missingness rate will indeed be r_{mis} . The HMM has a unique stationary state that we use as initial state, since the Markov chain is irreducible. For the missing value rates r_{mis} that we consider we have $p = \frac{1}{2}$, so the Markov chain is also aperiodic, hence the resulting missingness process is stationary, ergodic, and mixing.

Both of the missingness processes that we consider are independent of the obser-

7. LEARNING WITH MISSING VALUES

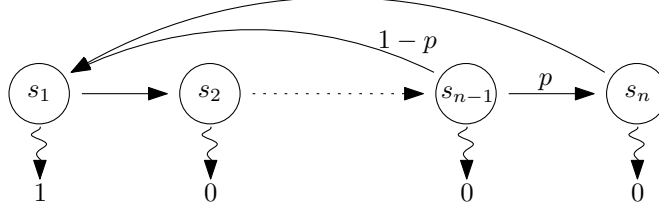


Figure 13: The HMM state transition structure describing the regular missingness process. The HMM has $n = \lceil \frac{1}{r_{\text{mis}}} \rceil$ states, where state s_1 always emits 1 (missing), while all other states emit 0 (not missing). The transition probability from state s_{n-1} to s_n is set to $p = \frac{1}{r_{\text{mis}}} - n + 1$ to ensure a missingness rate of r_{mis} . The initial state is the unique stationary state.

vation process, so the missingness is MCAR. Also, both missingness processes are stationary, ergodic, and mixing. Therefore, the combined input-output process governing the observed data with missing values will be stationary and ergodic. However, the missingness with regularly missing values is not strictly uncertain, and we will see how this affects the model estimation via Algorithm 13.

We will now explain how we perform the IO-OOM learning step of Algorithm 13 for the training data with missing values.

First we note that the training data is rather special, as it may be regarded as an input-output sequence with alphabets $\Sigma_O = \Sigma \cup \{\emptyset\}$ and $\Sigma_I = \{0, 1\}$, or equivalently just as a sequence with missing values over the alphabet $\Sigma \cup \{\emptyset\}$, since the symbols $(0, \emptyset)$ and $(1, x)$ for $x \in \Sigma$ cannot occur. We take this into account by using $|\Sigma_O| := |\Sigma| + 1$ and $|\Sigma_I| := 1$ wherever these quantities are required in the learning algorithm, which is in the computation of $O_{\min}$ for selecting indicative and characteristic sequences, and in the computation of the minimum $M_{\hat{\pi}}$ for the variance estimates. Furthermore, we may simply use the fixed minimum $M_{\hat{\pi}} = \frac{M}{N^2}$ for the variance estimates as in the case of stochastic processes, which essentially assumes $\pi = 1$, which is not correct but often performs significantly better than using the minimum $M_{\hat{\pi}}$ computed from an estimate of π . Therefore, we will always compute two model estimates, one using a minimum variance $M_{\hat{\pi}}$ based on an estimate $\hat{\pi}$, and one based on setting $\hat{\pi} = 1$, and select the one that gives the higher likelihood on the training data, evaluated on the last at most 100,000 time steps. Clearly, this is only relevant if the variance estimates are taken into account, i.e., for weighted spectral learning, and this calls for improvement.

As indicative and characteristic words we select the at most 1024 words of highest occurrence *relevance* that have an occurrence relevance of at least $O_{\min}$, where $O_{\min}$

is computed according to the heuristic presented in Section 5.2.1, using $|\Sigma_I| = 1$ and $|\Sigma_O| = |\Sigma| + 1$. The occurrence relevance $\text{rel}(\bar{y})$ of a word $\bar{y}$ is basically the normal occurrence count $\#(\bar{y})$, but additionally we penalize the relevance if the word $\bar{y}$ contains missing values. Let $|\bar{y}^{\text{mis}}|$ denote the number of missing values in $\bar{y}$. We will then consider two possibilities. First, we may simply discard all words that contain missing values by using $\text{rel}(\bar{y}) = \#(\bar{y})\delta_{|\bar{y}^{\text{mis}}|=0}$. In this case we only consider words without missing values, and the estimators $\hat{f}(\bar{y})$ that treat missing values either as symbols or as wildcards coincide. Alternatively, we may use $\text{rel}(\bar{y}) = \#(\bar{y})|\Sigma|^{-|\bar{y}^{\text{mis}}|}$, which we employ when using the estimators $\hat{f}(\bar{y})$ that treat the missing values as symbols, as well as when using the modified estimators that treat the missing values as wildcards. Note that the word selection Algorithm 5 can be easily adapted to take this more general relevance function into account by viewing the suffix tree as a trie, i.e., considering individual positions instead of nodes, since positions instead of nodes are then considered in order of descending relevance. As long as we employ a setting of $L_{\min} = 0$, this does not change the runtime complexity.

Let us summarize the three cases how we deal with missing values in the data. First, we can *discard* words containing missing values altogether. Alternatively, we can merely penalize the selection of words containing missing values, in which case we consider two different estimators $\hat{f}(\bar{y})$. The standard estimator for input-output sequences treats missing values as *symbols*, while the modified estimator treats missing values as *wildcards*.

As target dimension we use the process rank whenever this is known. We have seen that using the process rank as target dimension generally works reasonably well, so that this should suffice to be able to compare the effects of the other settings that we vary. For the real-world benchmarks where the process rank is unknown, we use the dimension estimated on the training data without missing values for the respective training data length, as is given in the following Table 7. The reason we do not estimate the target

$ \bar{s} $	10^3	$10^{3.5}$	10^4	$10^{4.5}$	10^5	$10^{5.5}$	10^6	$\sim 10^{6.5}$
ECOLI	2	3	4	5	9	14	28	53
BIBLE27	10	17	47	93	147	204	279	372

Table 7: Target dimensions used for real-world benchmarks of various data lengths.

dimension from the numerical rank of the matrix $\hat{F}_{Y,X}$ is that we have observed this to give erratic results when the training data contains missing values. We do not know why this is the case and leave this as an open problem requiring further research.

We compare two spectral learning methods, namely the standard spectral learning

7. LEARNING WITH MISSING VALUES

Algorithm 6 and the weighted spectral learning Algorithm 8. However, as we have done previously, in the case of the weighted spectral learning we resort to the element-wise weighted spectral learning algorithm if the target dimension exceeds 192. The results using standard spectral learning are shown in Figure 14, and the results using weighted spectral learning are shown in Figure 15.

7.3.1 Results

Let us first examine the results of standard spectral learning in the case of randomly missing values. We observe that for all methods of dealing with the missing values and all missingness rates, the estimated models appear to generally converge to a correct model when given sufficient training data, except for the ECOLI benchmark, where the quality of estimated models actually deteriorates for training data exceeding $10^{5.5}$ symbols. While we have not examined in detail why this happens, this may well have to do with the fact that we are using a target dimension optimized for data without missing values for the real-world benchmarks, which increases considerably in the ECOLI case for data lengths greater than $10^{5.5}$ and may be too large when values are missing.

Furthermore, we observe that standard spectral learning for randomly missing values mostly works best if the missing values are simply discarded. Treating missing values as wildcards often gives comparable results, but treating missing values as symbols does not work as well. We presume that the estimates for words containing missing values have a significantly lower precision, especially when the missingness rate is low and the missing values are treated as symbols, which then causes problems for the standard spectral learning algorithms. Indeed, we will see that this changes when considering weighted spectral learning. However, discarding occurrences containing missing values will necessarily discard a large amount of the available training data, so especially for higher missingness rates, we expect this to be an inefficient way of dealing with missing values. In fact, for the BIBLE27 benchmark with a missingness rate of $\frac{1}{2.5}$, discarding missing values gives the worst performance.

Next, let us examine what changes if we use weighted spectral learning instead but still consider randomly missing values. In this case, all methods for dealing with missing values generally give similar performance. Only on the real-world benchmarks ECOLI and BIBLE27 with a large missingness rate of $\frac{1}{2.5}$ we can see that discarding missing values gives significantly worse performance than the other methods. Furthermore, the performance in the case of missing values is slightly closer to the performance without missing values than it is when using standard spectral learning. Usually this means

that the performance is improved by using weighted spectral learning over standard spectral learning, but the BIBLE27 benchmark does not fit this pattern, since here standard spectral learning outperforms weighted learning even without missing values. Finally, weighted spectral learning with missing values works well also on the ECOLI benchmark, where standard spectral learning failed for larger amounts of training data. Weighted spectral learning, therefore, gives us more freedom in how to deal with missing values, while at the same time generally improving the performance compared to standard spectral learning.

Before we turn to the results for regularly missing values, let us explain a general problem that occurs in the case of regularly missing values. Namely, if values are missing regularly at a rate of r_{mis} , then at least every $\lceil \frac{1}{r_{\text{mis}}} \rceil$ time steps a value will be missing. Therefore, consistent estimates for $\hat{f}(\bar{x}z\bar{y})$ with $z \in \Sigma$ can only be obtained for words $\bar{x}$ and $\bar{y}$ that contain no missing values if $|\bar{x}z\bar{y}| < \lceil \frac{1}{r_{\text{mis}}} \rceil$, since otherwise the word $\bar{x}z\bar{y}$ simply does not occur in the training data. The reason we can have inconsistent estimates at all is because we have violated the requirement that the missingness must be strictly uncertain. This has two consequences:

First, if we allow only indicative and characteristic words that contain no missing values, then this places a bound on the rank of processes that can be estimated correctly by the general learning Algorithm 4. For example, if we consider a missingness rate of $r_{\text{mis}} = \frac{1}{4.5}$, then most processes of rank exceeding $2|\Sigma| + 2$ can no longer be estimated correctly. The reason is that most processes of rank d will have an operator τ_z of full rank d , which can only be estimated correctly via the learning Equations (2) if $\text{rank}(\hat{F}_{zY,X}) \geq d$. But only entries $\hat{f}(\bar{x}z\bar{y})$ for words $\bar{x}$ and $\bar{y}$ of length $|\bar{x}| + |\bar{y}| \leq 3$ can be estimated correctly, which are the entries where $\bar{x} = \varepsilon$ or $(\bar{x}, \bar{y}) \in \Sigma \times \Sigma^{\leq 2}$ or vice versa, so all other entries must be ignored, i.e., we may assume them to have an arbitrary value. It is easy to see that we can always fill the unknown entries in such a way that the resulting matrix $\hat{F}_{zY,X}$ has rank at most $2|\Sigma| + 2$.

Secondly, as soon as we have any entry in one of the estimated matrices $\hat{F}_{Y,X}$ or $\hat{F}_{zY,X}$ that we cannot estimate consistently, then this will introduce an error into the model estimated by standard spectral learning, i.e., standard spectral learning is no longer consistent. For instance, in the case of a missingness rate of $r_{\text{mis}} = \frac{1}{4.5}$, any typical selection of indicative and characteristic words X and Y where there are $\bar{x} \in X$ and $\bar{y} \in Y$ for words $\bar{x}, \bar{y} \in \Sigma^*$ such that $|\bar{x}| + |\bar{y}| \geq 4$ will run into this problem. It may be possible to circumvent this problem by using a creative selection strategy for indicative and characteristic words that is tailored to this case, for instance enforcing $X \subset \Sigma^{\leq 3}$ and $Y \subset \emptyset \Sigma^4$, but we do not consider this here. However, the weighted

7. LEARNING WITH MISSING VALUES

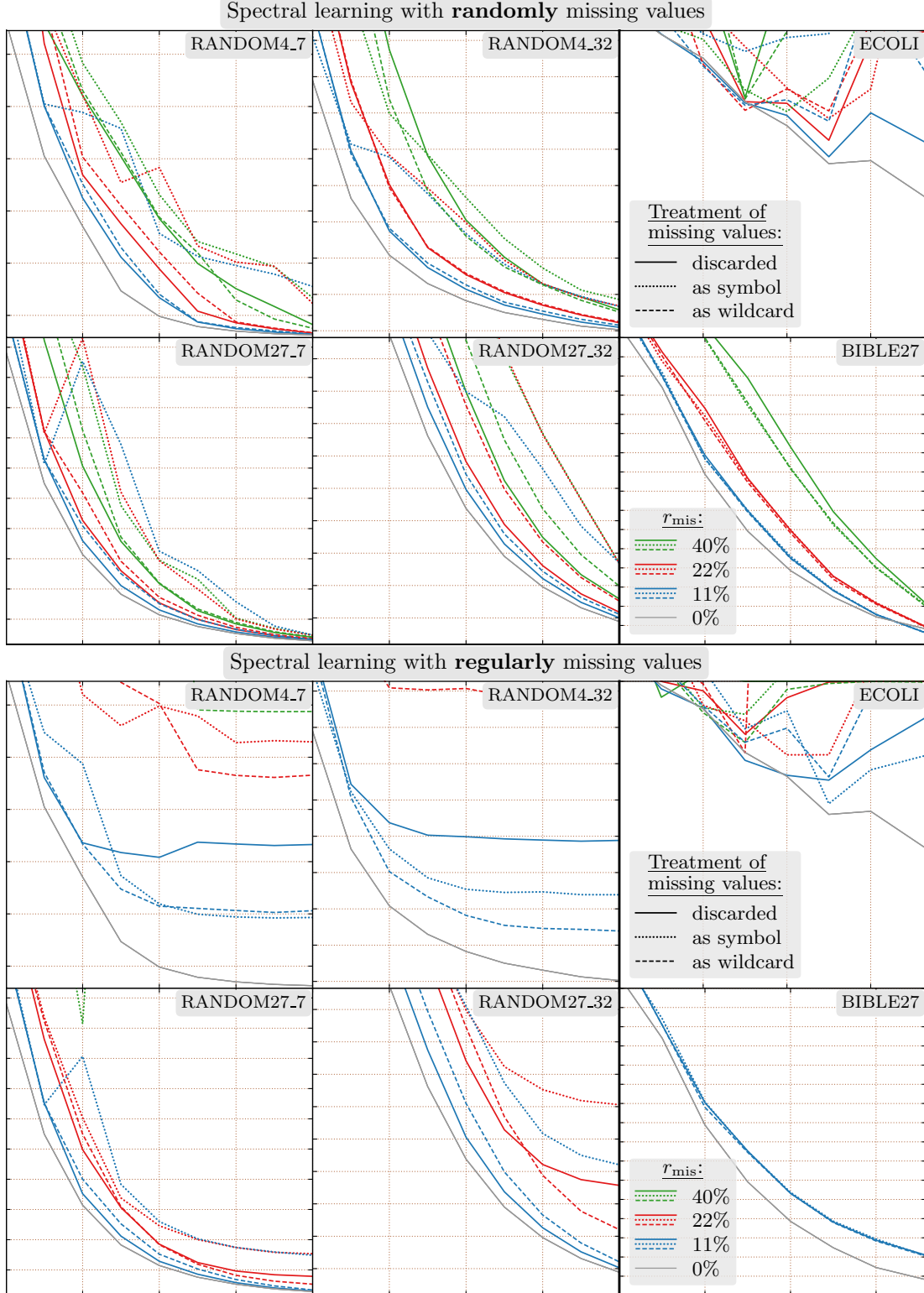


Figure 14: Evaluation of spectral learning on the benchmarks for stochastic processes with randomly or regularly missing values at missingness rates of $r_{\text{mis}} \in \{\frac{1}{9.5}, \frac{1}{4.5}, \frac{1}{2.5}\}$. Indicative and characteristic words are the ≤ 1024 words of highest relevance $\geq O_{\min}$ that either *discards* or penalizes words with missing values. In the latter case, estimators $\hat{f}(\bar{y})$ are considered that either treat missing values as *symbols* or as *wildcards*.

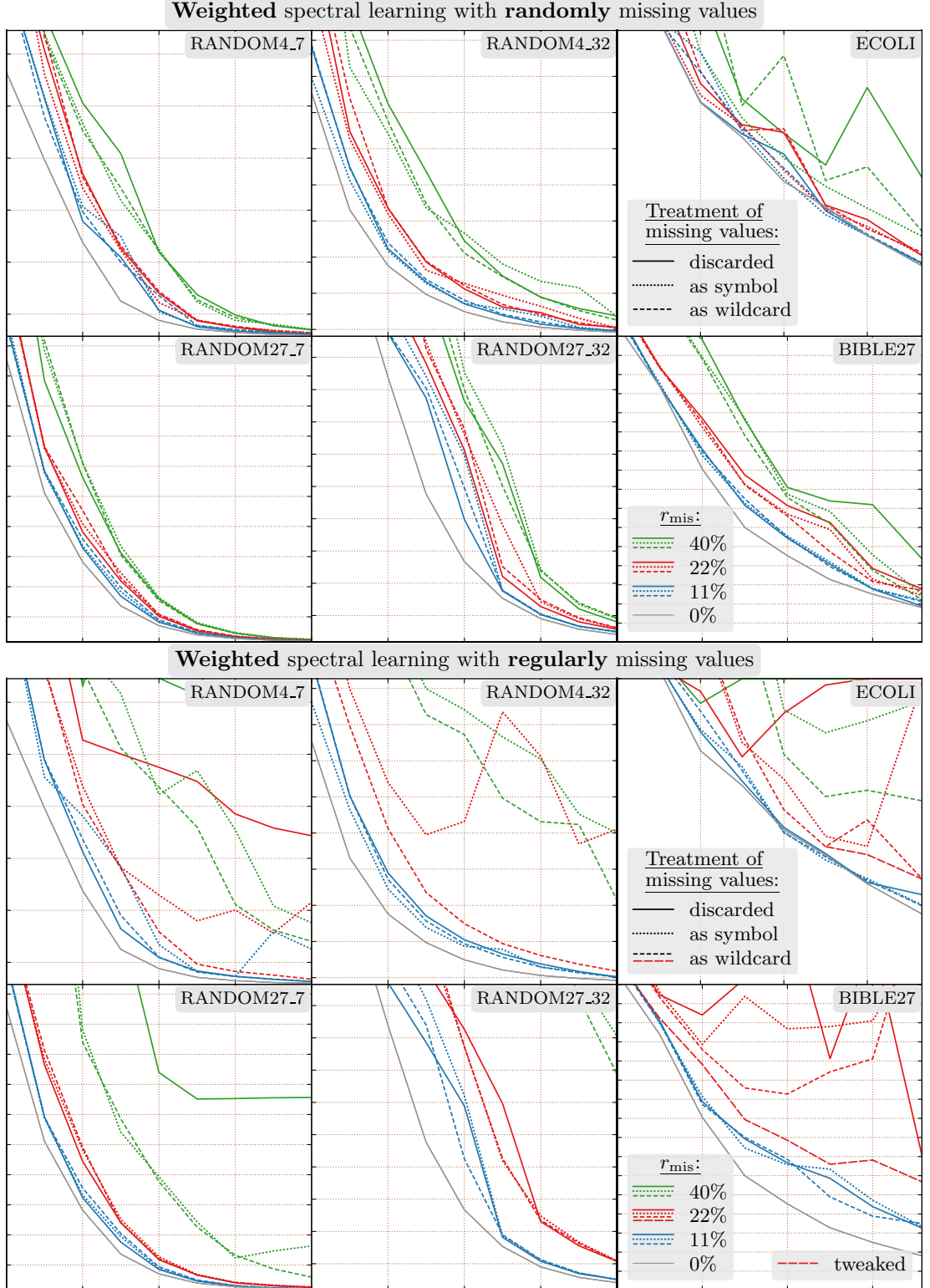


Figure 15: Evaluation of weighted spectral learning on the benchmarks for stochastic processes with randomly or regularly missing values at missingness rates of $r_{\text{mis}} \in \{\frac{1}{9.5}, \frac{1}{4.5}, \frac{1}{2.5}\}$. Indicative and characteristic words are the ≤ 1024 words of highest relevance $\geq O_{\min}$ that either *discards* or penalizes words with missing values. In the latter case, estimators $\hat{f}(\bar{y})$ are considered that either treat missing values as *symbols* or as *wildcards*.

7. LEARNING WITH MISSING VALUES

spectral learning Algorithm 8 should be able to cope with individual inconsistent estimates $\hat{f}(\bar{x}z\bar{y})$ that are based on zero occurrences in the training data by automatically assigning them a very small weight. It will therefore be interesting to see if weighted spectral learning is able to estimate correct models in cases where standard spectral learning fails.

Let us now examine the results in the case of regularly missing values. We first focus only on the strategy to discard missing, where only words without missing values are used as indicative and characteristic words. In this case, no correct models can be learned by standard or weighted spectral learning in the case of a missingness rate of $r_{\text{mis}} = \frac{1}{2.5}$ for any benchmark and in the case of a missingness rate of $r_{\text{mis}} = \frac{1}{4.5}$ for the benchmark RANDOM4.27 as well as the real-world benchmarks. This is likely due to the bound imposed on the rank of learnable processes discussed above. In the case of the RANDOM27.7 and RANDOM27.32 benchmarks and a missingness rate of $r_{\text{mis}} = \frac{1}{4.5}$, weighted spectral learning appears to be able to learn a correct model, while standard spectral learning clearly fails for the RANDOM27.32 benchmark. And considering a missingness rate of $r_{\text{mis}} = \frac{1}{9.5}$, weighted spectral learning is able to learn correct models for all benchmarks, while standard spectral learning works only for the benchmarks with an alphabet size of $|\Sigma| = 27$ and fails for the benchmarks with a smaller alphabet size of $|\Sigma| = 4$. The cases where standard spectral learning fails to learn a correct model but weighted spectral learning succeeds clearly cannot be due to some intrinsic bound on the rank of processes that can be learned, and are therefore likely due to the errors introduced by inconsistent estimate for entries in the matrices $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$.

We now observe what changes if indicative and characteristic words containing missing values are allowed. We first note that in the case of randomly missing values, as for the case of regularly missing values, treating missing values as wildcards gives at least as good and often substantially better results than treating missing values as symbols for standard as well as weighted spectral learning. This is what we had expected of these estimators, and we will therefore turn our attention to the results where missing values are treated as wildcards. As soon as we allow indicative and characteristic words containing missing values, there is no longer any principled bound on the model complexity that can be estimated. And indeed, it appears that in the case of weighted spectral learning models can now be estimated correctly in all cases of regularly missing values with a missingness rate of $r_{\text{mis}} = \frac{1}{4.5}$ (if we consider the “tweaked” curves that we will explain subsequently) and even in the cases RANDOM4.7, RANDOM4.32, and RANDOM27.7 with a missingness rate of $r_{\text{mis}} = \frac{1}{2.5}$. In the case of standard spectral

learning, the inclusion of indicative and characteristic words containing missing values does improve the results in the cases where no consistent estimate was possible without such words – and only in these cases. However, we can also see that the model estimation remains inconsistent in all of these cases.

Our interpretation is that the inclusion of words containing missing values yields a better exploitation of the available training data and overcomes the problem of a limit on the rank of processes that can be learnt, possibly at the cost of including matrix entries with lower precision, but cannot prevent the problem of inconsistent estimates entering the matrices $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$. Such inconsistent estimates, which introduce a fixed error into the matrices $\hat{F}_{Y,X}$ and $\hat{F}_{zY,X}$ as more training data is available, then cause the learning performance of the standard spectral learning algorithm to plateau at a sub-optimal level. Weighted spectral learning, however, can still obtain correct model estimates in these cases, apparently because it can ignore such inconsistent matrix entries by automatically assigning them very small weights.

Finally, we compare the performance in the case of regularly missing values to the performance in the case of randomly missing values for the weighted spectral learning algorithm when treating missing values as wildcards. In the case of a missingness rate of $r_{\text{mis}} = \frac{1}{9.5}$, the results are almost the same, except for the BIBLE27 benchmark, where the performance is slightly worse for regularly missing values. In the case of a missingness rate of $r_{\text{mis}} = \frac{1}{2.5}$, the performance is clearly worse for regularly missing values. And for a missingness rate of $r_{\text{mis}} = \frac{1}{4.5}$, the results are comparable on the synthetic benchmarks, but slightly worse on the real-world benchmarks, which even require employing the following “tweaked” settings in the case of regularly missing values:

For the ECOLI benchmark, the “tweak” consists of selecting the model learned for a training data length of 10^6 using a minimum variance $M_{\hat{\pi}}$ computed as in the case of stochastic processes, i.e., fixing $\hat{\pi} := 1$, instead of the model using a minimum variance $M_{\hat{\pi}}$ based on an estimate $\hat{\pi}$, which is otherwise selected due to a higher likelihood on the training data. This highlights the importance of the minimum variance parameter $M_{\hat{\pi}}$ and shows that the crude method to select the parameter based on higher likelihood on training data may be insufficient. For the BIBLE27 benchmark, we see that weighted spectral learning for a missingness rate of $r_{\text{mis}} = \frac{1}{4.5}$ and treating missing values as wildcards gives a performance that deteriorates as the amount of training data increases. We have seen such behavior before, and assume this is due to a poorly selected target dimension setting. Therefore, we have obtained the “tweaked” curve by using lower target dimension settings of $\{8, 16, 32, 64, 96, 128, 160, 192\}$ for the respective training

7. LEARNING WITH MISSING VALUES

data lengths, which are set based on intuition and not further optimized. This highlights the importance of selecting a suitable target dimension, which unfortunately in the case of data with missing values is currently an open problem.

We summarize the main findings as follows.

Conclusion 71. *Algorithm 13 using standard spectral learning works reliably only in the case of strictly uncertain missingness, i.e., here in the simple case of randomly missing values, and best when dealing with missing values by discarding them or treating them as wildcards. The performance is generally improved when using weighted spectral learning instead. Moreover, weighted spectral learning can also deal with regularly missing values, where the missingness is not strictly uncertain, if missing values are treated as wildcards. We therefore recommend to use Algorithm 13 together with weighted spectral learning and to treat missing values as wildcards for the most general applicability and overall best results.*

However, especially for real-world processes of possibly high rank, it is crucial to find a suitable setting for the target dimension, which is currently an open problem when dealing with missing values.

Conclusion

We have already included discussions and conclusions at the relevant places throughout the thesis, so that we will use this final conclusion chapter to point out several avenues for further research.

In Section 5.4.3 we showed how to extend the spectral learning algorithm to take weights into account that reflect the precision of the individual estimates $\hat{f}(\bar{x})$ for the entries in the system matrix $\hat{F}_{Y,X}$. In Section 5.1.1 we explained how to obtain variance estimates $\hat{V}[\hat{f}(\bar{x})]$ for this purpose. We employed several tricks for the variance estimates from Remark 39 that are required to make the estimates suitable for the weighted spectral learning algorithms. Most importantly, the variance estimates must not be zero, as their inverse is used in the weighted spectral learning algorithms. However, this poses a dilemma, as we wish to assign small variance estimates to very precise estimates $\hat{f}(\bar{x})$ that are based on many occurrences in the training data, while using a too small variance causes numerical issues for the weighted learning algorithm. Further insight into setting the parameters M and K that control the minimum variance and the strength of the Bayesian prior used for the variance estimation would be desirable. In particular, the situation in the case of weighted learning for data with missing values is unsatisfactory, where we need to resort to a crude form of cross-validation to optimize the minimum variance $M_{\hat{\pi}}$.

Possibly related to this, the estimation of the target dimension based on the numerical rank of the matrix $\hat{F}_{Y,X}$, which also uses variance estimates for the entries in $\hat{F}_{Y,X}$, requires further research especially in the case of data containing missing values, where the approaches from Section 5.3.1 currently fail. We speculate that the variance estimates from Section 5.1.1 may have problems dealing with extreme cases, where the estimates $\hat{f}(\bar{x})$ are either close to zero or one, or based only on very few counts, and

8. CONCLUSION

that such entries may occur much more frequently in the matrix $\hat{F}_{Y,X}$ when dealing with missing values. Interestingly, however, the model estimation using weighted spectral learning that is also based on these variance estimates does perform well even for data with missing values. In any case, this calls for further investigation.

Instead of improving the variance estimates, a complementary approach is to make the weighted spectral learning algorithms more numerically robust. A possible and standard method would be to include some form of regularizer for the required weighted linear regression problems. In fact, this might also be a way to address the strange behavior of the weighted spectral learning algorithm that we observe in situations where the target dimension is set too low for the complexity of the available data.

In general, there currently is a lack of understanding what the (weighted) spectral learning algorithm does if the target dimension is set lower than the process rank, even in the idealized case of a perfectly estimated system matrix. The intuition is that a best lower-rank approximation is learned, but our empirical results indicate that this may not be the case, at least not with regard to the cross-entropy. Alternatively, one may ask the question of how to obtain a lower-rank approximation to a given model that minimizes the cross-entropy. Some progress in this direction has recently been made by Kulesza et al. [2014, 2015a].

A final possible refinement to the current weighted spectral learning is to obtain the matrices $\hat{\tau}_z$ in the last step by solving $\hat{\tau}_z \hat{A} \approx \hat{A}_z$ as a weighted total least squares (WTLS) problem instead of a GLS problem. This additionally takes the precision of the entries in the matrix $\hat{A}$ into account, which are currently viewed as error-free. However, solving WTLS is computationally expensive. Also, we have experimented with this and presented initial results [Thon, 2013], but have found this to suffer from numerical issues. In fact, the TLS solution to a regression problem is in some sense a *deregularized* version of the corresponding OLS solution, so we have concluded that using TLS certainly requires the use of additional regularization. It is currently unclear if using the more involved WTLS methods in place of GLS can improve the performance of the weighted learning algorithm at all.

A long-standing open problem has been how to learn (IO)-OOM models for (controlled) processes from non-stationary data. As we have seen, this is needed by our approach to learn OOMs from data with missing values if the missingness is no longer MCAR, i.e., depends on the observation process. Recent progress has been made in the case of learning OOMs for non-stationary stochastic processes [Wu and Noé, 2016]. A careful analysis of these results and extension to the case of non-stationary controlled processes or even just the special case of learning OOM from data with missing values

would be a very valuable contribution. Related to this is the question which conditions are really necessary to obtain a consistent model estimation via the time-average estimates from Remark 37. We currently require the process to be stationary and ergodic but know that these requirements are too strong.

Last, but most importantly, it would be exciting to see real-world applications of the models and learning methods presented in this thesis, as it is only really possible to assess the utility of the various approaches in the context of a concrete application.

8. CONCLUSION

References

- Naoki Abe and Manfred K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260, 1992.
- Animashree Anandkumar, Daniel Hsu, and Sham M. Kakade. A method of moments for mixture models and hidden Markov models. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *Proceedings of the 25th Annual Conference on Learning Theory (COLT 2012)*, volume 23 of *JMLR Workshop & Conference Proceedings*, pages 33.1–33.34, 2012.
- Animashree Anandkumar, Rong Ge, Daniel J Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014.
- Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.
- Raphaël Bailly. Quadratic weighted automata: Spectral algorithm and likelihood maximization. In Chun-Nan Hsu and Wee Sun Lee, editors, *Proceedings of the 3rd Asian Conference on Machine Learning (ACML 2011)*, volume 20 of *JMLR Workshop & Conference Proceedings*, pages 147–163, 2011.
- Raphaël Bailly, François Denis, and Liva Ralivola. Grammatical inference as a principal component analysis problem. In Andrea Pohoreckýj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, volume 382 of *ACM Proceedings*, pages 33–40, 2009.
- Raphaël Bailly, Xavier Carreras, and Ariadna Quattoni. Unsupervised spectral learning of finite state transducers. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 800–808. Curran Associates, Inc., 2013.

REFERENCES

- Borja Balle and Mehryar Mohri. Spectral learning of general weighted automata via constrained matrix completion. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pages 2168–2176, 2012.
- Borja Balle, Ariadna Quattoni, and Xavier Carreras. A spectral learning algorithm for finite state transducers. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference (ECML/PKDD 2011), Proceedings, Part I*, volume 6911 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2011.
- Borja Balle, Ariadna Quattoni, and Xavier Carreras. Local loss optimization in operator models: A new insight into spectral learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*. icml.cc / Omnipress, 2012.
- Borja Balle, Xavier Carreras, Franco M. Luque, and Ariadna Quattoni. Spectral learning of weighted automata – a forward-backward perspective. *Machine Learning*, 96(1-2):33–63, 2014.
- Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. On the applications of multiplicity automata in learning. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS 1996)*, pages 349–358. IEEE Computer Society, 1996.
- Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *Journal of the ACM*, 47(3):506–530, 2000.
- Timothy Bell, Matt Powell, and Ross Arnold. The Canterbury corpus. <http://corpus.canterbury.ac.nz>, 1997.
- Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. In Maurizio A. Bonuccelli, Pierluigi Crescenzi, and Rossella Petreschi, editors, *Proceedings of the 2nd Italian conference on Algorithms and Complexity (CIAC 1994)*, volume 778 of *Lecture Notes in Computer Science*, pages 54–62. Springer, 1994.
- Francesco Bergadano, Dario Catalano, and Stefano Varricchio. Learning sat- k -DNF formulas from membership queries. In Gary L. Miller, editor, *Proceedings of the 28th*

REFERENCES

- Annual ACM Symposium on Theory of Computing (STOC 1996)*, pages 126–130. ACM, 1996.
- Jean Berstel, Jr. and Christophe Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1988.
- Byron Boots and Geoffrey J. Gordon. Predictive state temporal difference learning. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, pages 271–279. MIT Press, 2010.
- Byron Boots and Geoffrey J. Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 2011)*. AAAI Press, 2011.
- Byron Boots, Sajid M. Siddiqi, and Geoffrey J. Gordon. Closing the learning-planning loop with predictive state representations. In Wiebe van der Hoek, Gal A. Kaminka, Yves Lespérance, Michael Luck, and Sandip Sen, editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1369–1370. IFAAMAS, 2010.
- Byron Boots, Geoffrey J. Gordon, and Arthur Gretton. Hilbert space embeddings of predictive state representations. In Ann Nicholson and Padhraic Smyth, editors, *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI 2013)*, pages 92–101. AUAI Press, 2013.
- Michael Bowling, Peter McCracken, Michael James, James Neufeld, and Dana F. Wilkinson. Learning predictive state representations using non-blind policies. In William W. Cohen and Andrew Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, volume 148 of *ACM Proceedings*, pages 129–136, 2006.
- Leo Breiman et al. The individual ergodic theorem of information theory. *The Annals of Mathematical Statistics*, 28(3):809–811, 1957. Correction 31(3):809–810, 1960.
- Jack W. Carlyle and Azaria Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1):26–40, 1971.
- Anthony R. Cassandra. Tony’s POMDP page. <http://www.pomdp.org>, 1999.

REFERENCES

- Corinna Cortes and Mehryar Mohri. Context-free recognition with weighted automata. *Grammars*, 3(2/3):133–150, 2000.
- Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.
- François Denis and Yann Esposito. Learning classes of probabilistic automata. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the 17th Annual Conference on Learning Theory (COLT 2004)*, volume 3120 of *Lecture Notes in Computer Science*, pages 124–139. Springer, 2004.
- François Denis and Yann Esposito. On rational stochastic languages. *Fundamenta Informaticae*, 86(1):41–77, 2008.
- François Denis, Yann Esposito, and Amaury Habrard. Learning rational stochastic languages. In Gábor Lugosi and Hans-Ulrich Simon, editors, *Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006)*, volume 4005 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2006.
- Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 2009.
- Pierre Dupont, François Denis, and Yann Esposito. Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9):1349–1371, 2005.
- Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- Eyal Even-Dar, Sham M. Kakade, and Yishay Mansour. Planning in POMDPs using multiplicity automata. In Fahiem Bacchus and Tommi Jaakkola, editors, *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI 2005)*, pages 185–192. AUAI Press, 2005.
- Michel Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 53: 197–222, 1974.
- Edgar J. Gilbert. On the identifiability problem for functions of finite Markov chains. *The Annals of Mathematical Statistics*, 30(3):688–697, 1959.

REFERENCES

- Hadrien Glaude, Cyrille Enderli, and Olivier Pietquin. Non-negative spectral learning for linear sequential systems. In Sabri Arik, Tingwen Huang, Weng Kin Lai, and Qingshan Liu, editors, *Proceedings of the 22nd International Conference on Neural Information Processing (ICONIP 2015), Part II*, volume 9490 of *Lecture Notes in Computer Science*, pages 143–151. Springer, 2015.
- Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.
- Robert M. Gray. *Probability, Random Processes, and Ergodic Properties*. Springer, 1988.
- Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.
- William L. Hamilton, Mahdi M. Fard, and Joelle Pineau. Modelling sparse dynamical systems with compressed predictive state representations. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, volume 28 of *JMLR Workshop & Conference Proceedings*, pages 178–186, 2013.
- Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.
- Alex Heller. On stochastic processes derived from Markov chains. *The Annals of Mathematical Statistics*, 36(4):1286–1291, 1965.
- Daniel Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden Markov models. In Sanjoy Dasgupta and Adam Klivans, editors, *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009)*, 2009.
- Hisashi Ito. *An Algebraic Study of Discrete Stochastic Systems*. Unpublished doctoral dissertation, University of Tokyo, Bunkyo-ku, Tokyo, 1992.
- Hisashi Ito, Shun ichi Amari, and Kingo Kobayashi. Identifiability of hidden Markov information sources and their minimum degrees of freedom. *IEEE Transactions on Information Theory*, 38(2):324–333, 1992.

REFERENCES

- Masoumeh T. Izadi and Doina Precup. Point-based planning for predictive state representations. In Sabine Bergler, editor, *Proceedings of the 21st Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence (Canadian AI 2008)*, volume 5032 of *Lecture Notes in Computer Science*, pages 126–137. Springer, 2008.
- Herbert Jaeger. Observable operator models and conditioned continuation representations. Arbeitspapiere der GMD 1043, GMD Forschungszentrum Informationstechnik, Sankt Augustin, Germany, 1997.
- Herbert Jaeger. Discrete-time, discrete-valued observable operator models: a tutorial. Technical Report 42, GMD Forschungszentrum Informationstechnik, Sankt Augustin, Germany, 1998.
- Herbert Jaeger. Modeling and learning continuous-valued stochastic processes with OOMs. GMD Report 102, GMD Forschungszentrum Informationstechnik, Sankt Augustin, Germany, 2000a.
- Herbert Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000b.
- Herbert Jaeger, MingJie Zhao, and Andreas Kolling. Efficient estimation of OOMs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 555–562. MIT Press, 2006a.
- Herbert Jaeger, MingJie Zhao, Klaus Kretzschmar, Tobias Oberstein, Dan Popovici, and Andreas Kolling. Learning observable operator models via the ES algorithm. In Simon Haykin, José C. Príncipe, Terrence J. Sejnowski, and John McWhirter, editors, *New Directions in Statistical Signal Processing: From Systems to Brains*, Neural Information Processing, chapter 14, pages 417–464. MIT Press, Cambridge, MA, USA, 2006b.
- Michael R. James and Satinder P. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In Carla E. Brodley, editor, *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, volume 69 of *ACM Proceedings*, pages 53–60, 2004.
- Michael R. James and Satinder P. Singh. Planning in models that combine memory with predictive representations of state. In Manuela M. Veloso and Subbarao Kambham-

REFERENCES

- pati, editors, *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 987–992. AAAI Press, 2005.
- Michael R. James, Satinder Singh, and Michael L. Littman. Planning with predictive state representations. In *Proceedings of the 3rd International Conference on Machine Learning and Applications (ICMLA 2004)*, pages 304–311. IEEE Computer Society, 2004.
- Nan Jiang, Alex Kulesza, and Satinder P Singh. Improving predictive state representations via gradient descent. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 1709–1715. AAAI Press, 2016.
- Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2): 99–134, 1998.
- Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS 1996)*, pages 66–75. IEEE Computer Society, 1997.
- Klaus Kretzschmar. Learning symbol sequences with observable operator models. GMD Report 161, GMD Forschungszentrum Informationstechnik, Sankt Augustin, Germany, 2003.
- Alex Kulesza, N Raj Rao, and Satinder Singh. Low-rank spectral learning. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, volume 33 of *JMLR Workshop & Conference Proceedings*, pages 522–530, 2014.
- Alex Kulesza, Nan Jiang, and Satinder Singh. Low-rank spectral learning with weighted loss functions. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS 2015)*, volume 38 of *JMLR Workshop & Conference Proceedings*, pages 517–525, 2015a.
- Alex Kulesza, Nan Jiang, and Satinder P Singh. Spectral learning of predictive state representations with insufficient statistics. In Blai Bonet and Sven Koenig, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, volume 15, pages 2715–2721. AAAI Press, 2015b.

REFERENCES

- Michael L. Littman, Richard S. Sutton, and Satinder P. Singh. Predictive representations of state. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, pages 1555–1561. MIT Press, 2001.
- Ivan Markovsky and Sabine Van Huffel. Left vs right representations for solving weighted low-rank approximation problems. *Linear Algebra and its Applications*, 422(2-3):540–552, 2007a.
- Ivan Markovsky and Sabine Van Huffel. Overview of total least-squares methods. *Signal Processing*, 87(10):2283–2302, 2007b.
- Peter McCracken and Michael H. Bowling. Online discovery and learning of predictive state representations. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18 (NIPS 2005)*. MIT Press, 2006.
- Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.
- Cristopher Moore and James P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237(1-2):275–306, 2000.
- Feliks Nüske, Hao Wu, Jan-Hendrik Prinz, Christoph Wehmeyer, Cecilia Clementi, and Frank Noé. Markov state models from short non-equilibrium simulations – analysis and correction of estimation bias. *The Journal of Chemical Physics*, 146(9):094104, 2017.
- Hiroyuki Ohnishi, Hiroyuki Seki, and Tadao Kasami. A polynomial time learning algorithm for recognizable series. *IEICE Transactions on Information and Systems*, E77-D(10):1077–1085, 1994.
- Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- Adrià Recasens and Ariadna Quattoni. Spectral learning of sequence taggers over continuous sequences. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Zelezny, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference (ECML/PKDD 2013), Proceedings, Part I*, volume 8188 of *Lecture Notes in Computer Science*, pages 289–304. Springer, 2013.

REFERENCES

- Matthew Rosencrantz, Geoffrey J. Gordon, and Sebastian Thrun. Learning low dimensional predictive representations. In Carla E. Brodley, editor, *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, volume 69 of *ACM Proceedings*, pages 695–702, 2004.
- Sam Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS 1997)*, pages 626–632. MIT Press, 1998.
- Donald B. Rubin. Inference and missing data. *Biometrika*, 63:581–592, 1976.
- Matthew Rudary and Satinder P. Singh. Predictive linear-Gaussian models of controlled stochastic dynamical systems. In William W. Cohen and Andrew Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, volume 148 of *ACM Proceedings*, pages 777–784, 2006.
- Matthew Rudary and Satinder P. Singh. Predictive linear-Gaussian models of stochastic dynamical systems with vector-value actions and observations. In *Proceedings of the 10th International Symposium on Artificial Intelligence and Mathematics (ISAIM 2008)*, 2008.
- Matthew Rudary, Satinder P. Singh, and David Wingate. Predictive linear-Gaussian models of stochastic dynamical systems. In Fahiem Bacchus and Tommi Jaakkola, editors, *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI 2005)*, pages 501–508. AUAI Press, 2005.
- Matthew R. Rudary and Satinder Singh. A nonlinear predictive state representation. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 855–862. MIT Press, 2003.
- Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978.
- Alexander Schönhuth and Herbert Jaeger. Characterization of ergodic hidden Markov sources. *Information Theory, IEEE Transactions on*, 55(5):2107–2118, 2009.
- M. Schuermans, I. Markovsky, P.D. Wentzell, and Sabine Van Huffel. On the equivalence between total least squares and maximum likelihood PCA. *Analytica Chimica Acta*, 544(1):254–267, 2005.

REFERENCES

- Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.
- Shaun Seaman, John Galati, Dan Jackson, and John Carlin. What is meant by “missing at random”? *Statistical Science*, 28(2):257–268, 2013.
- Amirreza Shaban, Mehrdad Farajtabar, Bo Xie, Le Song, and Byron Boots. Learning latent variable models by improving spectral solutions with exterior point method. In Marina Meila and Tom Heskes, editors, *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI 2015)*, pages 792–801. AUAI Press, 2015.
- Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- Sajid M. Siddiqi, Byron Boots, and Geoffrey J. Gordon. Reduced-rank hidden Markov models. In Yee Whye Teh and D. Mike Titterton, editors, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9 of *JMLR Workshop & Conference Proceedings*, pages 741–748, 2010.
- Satinder Singh, Michael R. James, and Matthew R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In Joseph Halpern, editor, *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI 2004)*, pages 512–519. AUAI Press, 2004.
- Satinder P. Singh, Michael L. Littman, Nicholas K. Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In Tom Fawcett and Nina Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, pages 712–719. AAAI Press, 2003.
- Le Song, Byron Boots, Sajid M. Siddiqi, Geoffrey J. Gordon, and Alex J. Smola. Hilbert space embeddings of hidden Markov models. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pages 991–998. Omnipress, 2010.
- Michael Thon. Input-output OOMs. Technical Report 16, Jacobs University, Bremen, Germany, May 2008.
- Michael Thon. Weighted spectral learning and the efficiency sharpening algorithm. Presentation at the Spectral learning workshop, NIPS, 2013. http://videlectures.net/nipsworkshops2013_thon_spectral_learning.

- Michael Thon. Toolkit for observable operator modeling. GitHub repository, 2017. <https://github.com/m7thon/tom/releases/tag/phd>.
- Michael Thon and Herbert Jaeger. Links between multiplicity automata, observable operator models and predictive state representations – a unified learning framework. *Journal of Machine Learning Research*, 16:103–147, 2015. URL <http://jmlr.org/papers/v16/thon15a.html>.
- Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622, 1999.
- Daniel R. Upper. *Theory and Algorithms for Hidden Markov Models and Generalized Hidden Markov Models*. PhD thesis, University of California at Berkeley, 1997.
- Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- P.D. Wentzell, D.T. Andrews, D.C. Hamilton, K. Faber, and B.R. Kowalski. Maximum likelihood principal component analysis. *Journal of Chemometrics*, 11(4):339–366, 1997.
- Eric W. Wiewiora. *Modeling Probability Distributions with Predictive State Representations*. PhD thesis, University of California, San Diego, 2008.
- David Wingate and Satinder P. Singh. Kernel predictive linear Gaussian models for nonlinear stochastic dynamical systems. In William W. Cohen and Andrew Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, volume 148 of *ACM Proceedings*, pages 1017–1024, 2006a.
- David Wingate and Satinder P. Singh. Mixtures of predictive linear Gaussian models for nonlinear, stochastic dynamical systems. In Anthony Cohn, editor, *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006)*. AAAI Press, 2006b.
- David Wingate and Satinder P. Singh. Exponential family predictive representations of state. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 1617–1624. MIT Press, 2008a.

REFERENCES

- David Wingate and Satinder P. Singh. Efficiently learning linear-linear exponential family predictive representations of state. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, volume 307 of *ACM Proceedings*, pages 1176–1183, 2008b.
- David Wingate, Vishal Soni, Britton Wolfe, and Satinder P. Singh. Relational knowledge with predictive state representations. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2035–2040. AAAI Press, 2007.
- Britton Wolfe and Satinder P. Singh. Predictive state representations with options. In William W. Cohen and Andrew Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, volume 148 of *ACM Proceedings*, pages 1025–1032, 2006.
- Britton Wolfe, Michael R. James, and Satinder P. Singh. Learning predictive state representations in dynamical systems without reset. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, volume 119 of *ACM Proceedings*, pages 980–987, 2005.
- Hao Wu and Frank Noé. Spectral learning of dynamic systems from nonequilibrium data. In D.D. Lee, M. Sugiyama, and U.V. Luxburg, editors, *Advances in Neural Information Processing Systems*, pages 4179–4187. Curran Associates, Inc., 2016.
- Hao Wu, Jan-Hendrik Prinz, and Frank Noé. Projected metastable Markov processes and their estimation with observable operator models. *The Journal of Chemical Physics*, 143(14):144101, 2015.
- Lotfi Asker Zadeh. The concept of system, aggregate, and state in system theory. In Lotfi Asker Zadeh and Elijah Polak, editors, *System Theory*, volume 8 of *Inter-University Electronics Series*, pages 3–42. McGraw-Hill, New York, 1969.
- MingJie Zhao and Herbert Jaeger. Norm observable operator models. *Neural Computation*, 22(7):1927–1959, 2010.
- MingJie Zhao, Herbert Jaeger, and Michael Thon. A bound on modeling error in observable operator models and an associated learning algorithm. *Neural Computation*, 21(9):2687–2712, 2009a.

REFERENCES

MingJie Zhao, Herbert Jaeger, and Michael Thon. Making the error-controlling algorithm of observable operator models constructive. *Neural Computation*, 21(12): 3460–3486, 2009b.