Automatica 95 (2018) 187–196

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Tensor network subspace identification of polynomial state space models☆

Kim Batselier *, Ching-Yun Ko, Ngai Wong

The Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 25 September 2017
Received in revised form 22 February 2018
Accepted 15 April 2018

Keywords:
Subspace methods
Tensors
MIMO
Identification methods
System identification
Linear/nonlinear models

ABSTRACT

This article introduces a tensor network subspace algorithm for the identification of specific polynomial state space models. The polynomial nonlinearity in the state space model is completely written in terms of a tensor network, thus avoiding the curse of dimensionality. We also prove how the block Hankel data matrices in the subspace method can be exactly represented by low rank tensor networks, reducing the computational and storage complexity significantly. The performance and accuracy of our subspace identification algorithm are illustrated by experiments, showing that our tensor network implementation identifies a seventh degree polynomial state space model around 20 times faster than the standard matrix implementation before the latter fails due to insufficient memory. The proposed algorithm is also robust with respect to noise and therefore applicable to practical systems.

1. Introduction

Linear time-invariant (LTI) systems (Kailath, 1980) are a very useful framework for describing dynamical systems and have consequently been applied in myriad domains. Parametric system identification deals with the estimation of parameters for a given model structure from a set of measured vector input–output pairs $(\boldsymbol{u}_0, \boldsymbol{y}_0), \ldots, (\boldsymbol{u}_{L-1}, \boldsymbol{y}_{L-1})$ and has been thoroughly studied in the 1980s and 1990s. Two important model structures for LTI systems are transfer function models and state space models, which can be converted into one another. The dominant framework for the estimation of transfer function models is prediction error and instrumental variables methods (Ljung, 1999; Söderström & Stoica, 1988), while state space models are typically estimated through subspace methods (Katayama, 2005; van Overschee & De Moor, 2012). Prediction error methods are iterative methods that estimate the transfer function parameters such that the resulting prediction errors are minimized. These iterative methods suffer from some disadvantages such as no guaranteed convergence, sensitivity of the result on initial estimates and getting stuck in a local minimum of the objective function. Subspace methods, on the other hand, are non-iterative methods that rely on numerical linear algebra operations such as the singular value decomposition (SVD) or QR decomposition (Golub & Loan, 1996) of particular block Hankel data matrices. Estimates found through subspace methods are often good initial guesses for the iterative prediction error methods.

The most general nonlinear extension of the discrete-time linear state space model is

$$\begin{aligned} \boldsymbol{x}_{t+1} &= f(\boldsymbol{x}_t, \boldsymbol{u}_t), \\ \boldsymbol{y}_t &= g(\boldsymbol{x}_t, \boldsymbol{u}_t), \end{aligned}$$

where $\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{y}_t$ are the state, input and output vectors at time t, respectively and $f(\cdot), g(\cdot)$ are nonlinear vector functions. By choosing different nonlinear functions $f(\cdot), g(\cdot)$ one effectively ends up with very different nonlinear state space models. A popular choice for the nonlinear functions is multivariate polynomials. The most general polynomial state space model, where $f(\cdot), g(\cdot)$ are multivariate polynomials in both the state and the input, is described in Paduart et al. (2010). Being the most general form implies that it has a large expressive power, enabling the description of many different kinds of dynamics. This, however, comes at the cost of having to estimate an exponentially growing number of parameters as the degree of the polynomials increases. Furthermore, the identification method relies on solving a highly nonlinear optimization problem using iterative methods.

In Kruppa, Pangalos, and Lichtenberg (2014), multilinear time invariant (MTI) systems are proposed. MTI systems are systems for which $f(\cdot), g(\cdot)$ are multivariate polynomial functions where each state or input variable is limited to a maximal degree of one. The

☆ This work was supported in part by the Hong Kong Research Grants Council under General Research Fund (GRF) Project 17246416. The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Antonio Vicino under the direction of Editor Torsten Söderström.

* Corresponding author.
E-mail addresses: kimb@eee.hku.hk (K. Batselier), cyko@eee.hku.hk (C.-Y. Ko), nwong@eee.hku.hk (N. Wong).

https://doi.org/10.1016/j.automatica.2018.05.015
0005-1098/

number of parameters of an m-input–p-output MTI system with n states is then $(n+p)2^{(n+m)}$, growing exponentially with the number of state and input variables. This curse of dimensionality is then effectively lifted with tensor methods. In this article, we propose the following polynomial state space model

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{A}\,\boldsymbol{x}_t + f(\boldsymbol{u}_t), \\ \boldsymbol{y}_t &= \boldsymbol{C}\,\boldsymbol{x}_t + g(\boldsymbol{u}_t), \end{aligned} \tag{1}$$

where both $f(\cdot), g(\cdot)$ are multivariate polynomials of total degree d. We then show that it is possible to identify these models using conventional subspace methods. The number of parameters that need to be estimated, however, will also grow exponentially with d. We then propose to use tensor networks to represent the polynomials $f(\cdot), g(\cdot)$ and modify the conventional MOESP (Multivariable Output Error State Space) subspace method to work for tensor networks. The main contributions of this article are:

(1) We extend LTI state space models to a specific class of polynomial state space models with a linear state sequence and a polynomial input relation.
(2) We modify the MOESP method (Verhaegen & Dewilde, 1992a,b) to utilize tensor networks for the identification of the proposed polynomial state space model.
(3) We prove in Theorem 4.1 that the block Hankel data matrices in subspace methods are exactly represented by low-rank tensor networks, thereby reducing the computational and storage complexity significantly.

The main outline of this article is as follows. First, we briefly discuss some tensor network preliminaries in Section 2. The proposed polynomial state space model is discussed in detail in Section 3. The development and implementation of our proposed tensor network subspace identification method is described in Section 4. The algorithm to simulate our proposed polynomial state space model in tensor network form is given in Section 5. Numerical experiments validate and demonstrate the efficacy of our tensor network subspace identification method in Section 6. All our algorithms were implemented in the MATLAB/Octave TNMOESP package and can be freely downloaded from https://github.com/kbatseli/TNMOESP. Finally, some conclusions and future work are formulated in Section 7.

2. Preliminaries

Most of the notation on subspace methods is adopted from Katayama (2005) and the notation on tensors from Batselier, Chen, and Wong (2017a, b) is also used. Tensors are multi-dimensional arrays that generalize the notions of vectors and matrices to higher orders. A d-way or dth-order tensor is denoted $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and hence each entry of $\mathcal{A}$ is determined by d indices $i_1, \ldots, i_d$. We use the MATLAB convention that indices start from 1, such that $1 \leq i_k \leq n_k\,(k = 1, \ldots, d)$. The numbers $n_1, n_2, \ldots, n_d$ are called the dimensions of the tensor. For practical purposes, only real tensors are considered. We use boldface capital calligraphic letters $\mathcal{A}, \mathcal{B}, \ldots$ to denote tensors, boldface capital letters $\boldsymbol{A}, \boldsymbol{B}, \ldots$ to denote matrices, boldface letters $\boldsymbol{a}, \boldsymbol{b}, \ldots$ to denote vectors, and Roman letters $a, b, \ldots$ to denote scalars. The elements of a set of d tensors, in particular in the context of tensor networks, are denoted $\mathcal{A}^{(1)}, \mathcal{A}^{(2)}, \ldots, \mathcal{A}^{(d)}$. The transpose of a matrix $\boldsymbol{A}$ or vector $\boldsymbol{a}$ are denoted $\boldsymbol{A}^T$ and $\boldsymbol{a}^T$, respectively. The unit matrix of order n is denoted $\boldsymbol{I}_n$. A matrix with all zero entries is denoted $\boldsymbol{O}$.

A very useful graphical representation of scalars, vectors, matrices and tensors is shown in Fig. 1. The number of unconnected edges of each node represents the order of the corresponding tensor. Scalars therefore are represented by nodes without any edges, while a matrix is represented by a node that has two edges. This

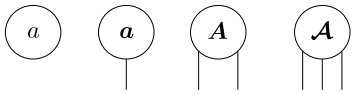

Fig. 1. Graphical depiction of a scalar a, vector $\boldsymbol{a}$, matrix $\boldsymbol{A}$ and 3-way tensor $\mathcal{A}$.

graphical representation allows us to visualize the different tensor networks and operations in this article in a very straightforward way. We also adopt the MATLAB notation regarding entries of tensors, e.g. $\boldsymbol{A}(:, 1)$ denotes the first column of the matrix $\boldsymbol{A}$. We now give a brief description of some required tensor operations. The generalization of the matrix–matrix multiplication to tensors involves a multiplication of a matrix with a d-way tensor along one of its d possible modes.

Definition 2.1 (*Kolda & Bader, 2009, p. 460*)**.** The k-mode product of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_k \times \cdots \times n_d}$ with a matrix $\boldsymbol{U} \in \mathbb{R}^{p_k \times n_k}$ is denoted $\mathcal{B} = \mathcal{A} \times_k \boldsymbol{U}$ and defined by

$$\mathcal{B}(i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_d) = \sum_{i_k=1}^{n_k} \boldsymbol{U}(j, i_k)\mathcal{A}(i_1, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_d), \tag{2}$$

with $\mathcal{B} \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times p_k \times n_{k+1} \times \cdots \times n_d}$.

For a $(d+1)$-way tensor $\mathcal{A} \in \mathbb{R}^{n \times m \times \cdots \times m}$ and vector $\boldsymbol{x} \in \mathbb{R}^m$, we define the short hand notation for the vector

$$\mathcal{A}\boldsymbol{x}^d := \mathcal{A} \times_2 \boldsymbol{x}^T \times_3 \cdots \times_{d+1} \boldsymbol{x}^T \in \mathbb{R}^n.$$

The Kronecker product will be repeatedly used to describe the polynomial nonlinearity.

Definition 2.2 (*Kronecker Product*)**.** If $\boldsymbol{B} \in \mathbb{R}^{m_1 \times m_2}$ and $\boldsymbol{C} \in \mathbb{R}^{n_1 \times n_2}$, then their Kronecker product $\boldsymbol{B} \otimes \boldsymbol{C}$ is the $m_1 n_1 \times m_2 n_2$ matrix

$$\begin{pmatrix} b_{11} & \cdots & b_{1m_2} \\ \vdots & \ddots & \vdots \\ b_{m_1 1} & \cdots & b_{m_1 m_2} \end{pmatrix} \otimes \boldsymbol{C} = \begin{pmatrix} b_{11}\boldsymbol{C} & \cdots & b_{1m_2}\boldsymbol{C} \\ \vdots & \ddots & \vdots \\ b_{m_1 1}\boldsymbol{C} & \cdots & b_{m_1 m_2}\boldsymbol{C} \end{pmatrix}. \tag{3}$$

Definition 2.3. The Khatri–Rao product $\boldsymbol{A} \odot \boldsymbol{B}$ between $\boldsymbol{A} \in \mathbb{R}^{n_1 \times p}$ and $\boldsymbol{B} \in \mathbb{R}^{n_2 \times p}$ is the matrix $\boldsymbol{C} \in \mathbb{R}^{n_1 n_2 \times p}$ with

$$\boldsymbol{C}(:, k) = \boldsymbol{A}(:, k) \otimes \boldsymbol{B}(:, k),\ (k = 1, \ldots, p).$$

Two common operations on tensors that we will use throughout this article are reshaping and a permutation of the indices.

Definition 2.4. We adopt the MATLAB/Octave reshape operator "reshape($\mathcal{A}$, $[n_1, n_2, n_3 \cdots]$)", which reshapes the d-way tensor $\mathcal{A}$ with column-wise ordering preserved into a tensor with dimensions $n_1 \times n_2 \times \cdots \times n_d$. The total number of elements of $\mathcal{A}$ must be the same as $n_1 \times n_2 \times \cdots \times n_d$.

Definition 2.5. We adopt the MATLAB/Octave permutation operator "permute($\mathcal{A}$, $\boldsymbol{p}$)", which rearranges the dimensions of $\mathcal{A}$ so that they are in the order specified by the vector $\boldsymbol{p}$. The resulting tensor has the same values of $\mathcal{A}$ but the order of the subscripts needed to access any particular element is rearranged as specified by $\boldsymbol{p}$. All the elements of $\boldsymbol{p}$ must be unique, real, positive, integer values.

Storing all entries of a d-way tensor with dimension size n requires n^d storage units and quickly becomes prohibitively large for increasing values of n and d. When the data in the tensor have redundancies, then more economic ways of storing the tensor exist in the form of tensor decompositions. The tensor decomposition