Automatica 84 (2017) 17–25

Contents lists available at ScienceDirect

Automatica

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

A Tensor Network Kalman filter with an application in recursive MIMO Volterra system identification☆

Kim Batselier [a], Zhongming Chen [b], Ngai Wong [a]

[a] The Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
[b] Department of Mathematics, School of Science, Hangzhou Dianzi University, Hangzhou 310018, China

ARTICLE INFO

Article history:
Received 17 October 2016
Received in revised form 6 February 2017
Accepted 1 May 2017

Keywords:
Volterra series
Tensors
Kalman filters
Identification methods
System identification
MIMO
Time-varying systems

ABSTRACT

This article introduces a Tensor Network Kalman filter, which can estimate state vectors that are exponentially large without ever having to explicitly construct them. The Tensor Network Kalman filter also easily accommodates the case where several different state vectors need to be estimated simultaneously. The key lies in rewriting the standard Kalman equations as tensor equations and then implementing them using Tensor Networks, which effectively transforms the exponential storage cost and computational complexity into a linear one. We showcase the power of the proposed framework through an application in recursive nonlinear system identification of high-order discrete-time multiple-input multiple-output (MIMO) Volterra systems. The identification problem is transformed into a linear state estimation problem wherein the state vector contains all Volterra kernel coefficients and is estimated using the Tensor Network Kalman filter. The accuracy and robustness of the scheme are demonstrated via numerical experiments, which show that updating the Kalman filter estimate of a state vector of length 10^9 and its covariance matrix takes about 0.007 s on a standard desktop computer in Matlab.

1. Introduction

After its publication in 1960, the Kalman filter (Kalman, 1960) was quickly adopted into the Apollo onboard guidance system (McGee & Schmidt, 1985) and has found many other applications ever since. The square-root filter is a more numerically stable implementation and was first developed by Potter and Stern (1963). It replaces the covariance matrix in the Kalman filter equations by its Cholesky factor, which is better conditioned. Over the next decade other numerically stable implementations, which also use Cholesky factors, were developed (Bierman, 1977; Carlson, 1973; Morf & Kailath, 1975). Extensions of the Kalman filter to nonlinear models are the Extended Kalman filter (EKF) (Jazwinski, 2007; Schmidt, 1976), the statistically linearized filter (SLF) (Gelb, 1974) and the unscented Kalman filter (UKF) (Julier & Uhlmann, 2004; Julier, Uhlmann, & Durrant-Whyte, 1995). All these filters turn out to be specific instances of Bayesian filters (Särkkä, 2013), where the Kalman filter solution emerges from the assumption that both the dynamic and measurement models are linear Gaussian.

Kalman filters have been applied in myriad different fields, with probably the most famous application being in navigational and guidance systems. Other applications are found in time series analysis, signal processing and econometrics. The Kalman filter is inherently limited by the length of the state vector that is to be estimated. For example, using a Kalman filter to estimate a state vector with a length n^d will quickly become intractable, even for moderate sizes of n and d. In this article, we explain how Tensor Networks (Cichocki, 2014; Orús, 2014) enable the estimation of exponentially long state vectors in a computationally efficient manner. The main paradigm used in the Tensor Network framework is to represent the exponentially long state vectors and their corresponding covariance matrices as tensors in a network. These tensors are called the Tensor Network cores and all computations of the Kalman filter are performed directly on the cores. We show in Section 4 that this reduces the computational complexity and storage cost from $O(n^d)$ to $O(dn)$.

One particularly well-suited application of the Tensor Network Kalman filter is the recursive identification of discrete-time multiple-input-multiple-output (MIMO) Volterra systems (Rugh, 1981; Wiener, Stratton, & Technology, 2013). These nonlinear systems have been extensively studied and applied in applications like speech modeling (Mumolo & Francescato, 1993), loudspeaker linearization (Kajikawa, 2008), nonlinear control (Doyle, Pearson, & Ogunnaike, 2002), active noise control (Tan & Jiang, 2001), modeling of biological and physiological systems (Korenberg & Hunter,

☆ This work was supported in part by the Hong Kong Research Grants Council under General Research Fund (GRF) Projects 7212315 and 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.

E-mail addresses: kim.batselier@eee.hku.hk (K. Batselier), czm183015@126.com (Z. Chen), nwong@eee.hku.hk (N. Wong).

http://dx.doi.org/10.1016/j.automatica.2017.06.019
0005-1098/

1996), nonlinear communication channel identification and equalization (Cheng & Powers, 2001; Fernandes, Mota, & Favier, 2010), distortion analysis (Wambacq & Sansen, 1998) and many others. Their applicability has been limited however to "weakly nonlinear systems", where the nonlinear effects play a non-negligible role but are dominated by the linear terms. This limitation is not inherent to the Volterra series themselves, as they can also represent strongly nonlinear dynamical systems, but is due to the exponentially growing number of Volterra kernel coefficients as the degree increases. Indeed, assuming a finite memory M, the dth-order response of a discrete-time single-input single-output (SISO) Volterra system is given by

$$y_d(t) = \sum_{k_1,\ldots,k_d=0}^{M-1} h_d(k_1,\ldots,k_d) \prod_{i=1}^{d} u(t-k_i),$$

where $y_d(t)$, $u(t)$ are the scalar output and input at time t respectively and the dth-order Volterra kernel $h_d(k_1,\ldots,k_d)$ is described by M^d numbers. For a multiple-input multiple-output (MIMO) Volterra system with p inputs the situation gets even worse, since the dth-order Volterra kernel for one particular output is characterized by $(pM)^d$ numbers. This exponential growth of the number of kernel coefficients is one particular example of the infamous *curse of dimensionality*.

In order to apply the Tensor Network Kalman filter to the problem of recursive system identification of MIMO Volterra systems, we first rewrite the MIMO Volterra system as a linear state space model of the Volterra kernel coefficients. The system identification problem is in this way converted into a state estimation problem. The linear state space description of SISO Volterra systems for the identification of its kernel coefficients has appeared in Weng and Barner (2006). The curse of dimensionality however limits the application of their method to low degree Volterra systems. After having converted the MIMO Volterra system into a linear state space mode, we present a Tensor Network description of MIMO Volterra systems (Batselier, Chen, & Wong, 2016). This description effectively enables the use of a Tensor Network Kalman filter to solve the state estimation problem. In contrast with the system identification method described in Batselier et al. (2016), the Kalman filter approach explicitly takes the effect of measurements noise into account. Furthermore, we derive how the Tensor Network cores are initialized without the explicit construction of the prohibitively large mean vectors and covariance matrices. In short, the main contributions of this article are

- the Kalman filter equations are rewritten as tensor equations to accommodate for the estimation of multiple state vectors at once,
- each of the Kalman filter tensor equations are computed in the Tensor Network format, resulting in a significant reduction of computational complexity and storage cost,
- the Tensor Network Kalman filter is applied for the recursive identification of MIMO Volterra systems.

The outline of this article is as follows. In Section 2 we give a brief overview of important tensor concepts and Tensor Network theory. The Tensor Network Kalman filter is derived in Section 3 and its implementation is discussed in Section 4. The MIMO Volterra Tensor Network framework from Batselier et al. (2016) is reviewed and the application of the Tensor Network Kalman filter to the system identification problem is discussed in Section 5. In Section 6, numerical experiments demonstrate the accuracy and computational efficiency of the Tensor Network Kalman filter when applied for recursive MIMO Volterra system identification. Matlab/Octave implementations of our algorithms are freely available from https://github.com/kbatseli/TNKalman.

2. Preliminaries

2.1. Tensor basics

Tensors in this article 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 of its entries $a_{i_1 i_2 \ldots i_d}$ is determined by d indices. We use the 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. A tensor is cubical if all its dimensions are equal. 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 $\boldsymbol{a}$ is denoted $\boldsymbol{A}^T$ and $\boldsymbol{a}^T$, respectively. The unit matrix of order n is denoted $\boldsymbol{I}_n$. The tensor with all zero entries is denoted $\mathcal{O}$. We also adopt the Matlab notation $\mathrm{diag}(\boldsymbol{a})$ for a diagonal matrix with entries a_i. Similarly, $\mathrm{diag}(\boldsymbol{A})$ denotes the diagonal entries of a matrix $\boldsymbol{A}$.

Good introductions to tensors in scientific computing and signal processing are Cichocki, Mandic, Phan, Caiafa, Zhou, and Zhao, et al. (2015) and Kolda and Bader (2009). The work in this article builds upon the tensor framework described in Batselier et al. (2016), in which a Tensor Network alternating linear scheme is derived for the identification of MIMO Volterra systems. Due to space limitation, we refer the reader to the discussion presented in Batselier et al. (2016) on basic tensor operations. The same notation and concepts will be used in this article. Additional important tensor operations for this article that are not described in Batselier et al. (2016) are given below.

Definition 2.1 (*Kolda & Bader, 2009, p. 462 (Khatri–Rao Product)*)**.** Given matrices $\boldsymbol{A} \in \mathbb{R}^{n \times l}$, $\boldsymbol{B} \in \mathbb{R}^{m \times l}$, then their Khatri–Rao product $\boldsymbol{A} \odot \boldsymbol{B} \in \mathbb{R}^{nm \times l}$ is defined as the column-wise Kronecker product

$$\boldsymbol{A} \odot \boldsymbol{B} := \begin{pmatrix} \boldsymbol{A}(:,1) \otimes \boldsymbol{B}(:,1), & \cdots & , \boldsymbol{A}(:,l) \otimes \boldsymbol{B}(:,l) \end{pmatrix},$$

where we used the Matlab-notation $\boldsymbol{A}(:,k)$ to denote the kth column of the matrix $\boldsymbol{A}$.

The Khatri–Rao product of two matrices $\boldsymbol{A}, \boldsymbol{B}$ hence corresponds with the matrix that contains the column-wise Kronecker product of $\boldsymbol{A}$ with $\boldsymbol{B}$. Similarly, an operation will be required where the Kronecker product is replaced with the outer product.

Definition 2.2. Given matrices $\boldsymbol{A} \in \mathbb{R}^{n \times l}$, $\boldsymbol{B} \in \mathbb{R}^{m \times l}$, then the tensor $\boldsymbol{A} \square \boldsymbol{B} \in \mathbb{R}^{n \times m \times l}$ is defined as the column-wise outer product such that

$$\boldsymbol{A} \square \boldsymbol{B}(:,:,k) := \boldsymbol{A}(:,k) \circ \boldsymbol{B}(:,k),$$

for $k = 1,\ldots,l$ and where $\circ$ denotes the outer product (Kolda & Bader, 2009, p. 458).

One can obtain $\boldsymbol{A} \square \boldsymbol{B}$ from reshaping the matrix $\boldsymbol{A} \odot \boldsymbol{B}$ into a 3-way tensor. Next we will provide the definition of the tensor Kronecker product, but before doing so, we first need to discuss multi-indices. A d-way tensor $\mathcal{A}$ is essentially a collection of numbers $a_{i_1 \cdots i_d}$, and there are therefore many ways to arrange the entries. These different arrangements are equivalent with the grouping of the indices into separate groups. Consider the case where the first k indices are grouped together into the multi-index $[i_1 i_2 \cdots i_k]$, keeping all remaining indices separate. This reduces the order of the tensor from d to $d - k + 1$. The multi-index $[i_1 i_2 \cdots i_k]$ is converted into a single linear index as

$$i_1 + (i_2 - 1)n_1 + \cdots + (i_k - 1)n_1 n_2 \ldots n_{k-1}.$$