Large-Scale Distributed Kalman Filtering via an Optimization Approach

Mathias Hudoba de Badyn* Mehran Mesbahi*
* William E. Boeing Department of Aeronautics and Astronautics, University of Washington, Seattle, WA 98195 USA (e-mails: {hudomath,mesbahi}@uw.edu).

Abstract: Large-scale distributed systems such as sensor networks, often need to achieve filtering and consensus on an estimated parameter from high-dimensional measurements. Running a Kalman filter on every node in such a network is computationally intensive; in particular the matrix inversion in the Kalman gain update step is expensive. In this paper, we extend previous results in distributed Kalman filtering and large-scale machine learning to propose a gradient descent step for updating an estimate of the error covariance matrix; this is then embedded and analyzed in the context of distributed Kalman filtering. We provide properties of the resulting filters, in addition to a number of applications throughout the paper.

© 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Machine learning, fast Kalman algorithms, state estimation, gradient methods

1. INTRODUCTION

The Kalman filter is an algorithm that uses the known dynamics of a system to remove noise from measurements of that system. When considering large-scale dynamical systems, implementation of the standard or extended Kalman filters can be computationally difficult. In such cases, the Kalman filter requires the inversion of very large matrices at each timestep. This may cause the Kalman filter to run slower than the dynamical process it is trying to measure, or to severely reduce the temporal resolution of the measurements.

There are many systems for which measurements are taken by a network of sensors and are of high dimension; examples of such systems can be found in Khan and Moura (2008) and Kutz (2013). Previous methods for circumventing this problem include decomposing the dynamical system being measured into several subsystems and distributing the subsystems over the sensor network as in Khan and Moura (2008), or using Monte-Carlo methods for estimating the error covariance, such as in Furrer and Bengtsson (2007).

Previously, Sutton (1992) proposed to modify the Kalman filter error covariance update with a gradient descent method for the purpose of minimizing memory consumption, albeit for a specific instantiation of a SISO linear system. In this paper, we extend the gradient descent algorithm for estimating the Kalman filter error covariance to the general MIMO linear system as a proposed solution to the problem of running a Kalman filter on a high-dimensional system. We improve the gradient descent using Nesterov acceleration and adaptive learning rate methods. Lastly, we apply the methods above to distributed Kalman filtering on a sensor network.

Distributed Kalman filtering seeks to estimate the state of a system by distributing the tasks of measuring the system and subsequently filtering the data to many agents, who then collectively assemble the state estimate. Such algorithms utilizing consensus to provide a global estimate were presented by Olfati-Saber (2005) and Olfati-Saber (2007), and later extended by Carli et al. (2008). Performance of distributed Kalman filters using graph-theoretic quantities were studied by Spanos et al. (2005).

The organization of the paper is as follows. In §2, we outline the mathematical notation and conventions we use. We discuss the gradient descent algorithm, Nesterov acceleration and adaptive learning rate methods in §3, and the distributed version of these algorithms with a relevant example in §4. The paper is summarized in §5.

2. MATHEMATICAL PRELIMINARIES

In this section, we lay out the mathematical notation used in this paper, and summarize the essential background on Kalman filtering.

Consider the noisy discrete-time controlled linear system

\[ x_{k+1} = A x_k + B u_k + \gamma_k w_k \]
\[ \hat{y}_k = C_k x_k + v_k, \]

where \( x_k \in \mathbb{R}^n \) denotes the state vector, \( u_k \in \mathbb{R}^m \) denotes the control vector, \( \gamma_k \in \mathbb{R}^p \) is the output vector or measurement, and \( w_k, v_k \) are Gaussian white noise vectors of appropriate dimensions with covariances \( Q_k \) and \( R_k \) respectively. The subscript \( k \) refers to the timestep. The random vectors \( w_k \) and \( v_k \) represent system disturbances and sensor noise respectively, and the matrix \( \gamma_k \) describes how the disturbance propagates into the system. Since most matrix quantities will have a subscript \( k \) denoting the timestep, we will refer to the \( ij \)th entry of a matrix \( A_k \) with bracketed superscripts: \( A_k^{(ij)} \). Similarly, the ith
entry of a vector $a_k$ is denoted $a_k(i)$, the $i$th column of a matrix $A_k$ is denoted $A_k[i]$, and the $i$th row is denoted $A_k(i)$.

A network is represented by a graph $G = (V, E)$ where $V$ is a set of nodes representing agents in the network, and $E$ is the set of edges $\{i, j\}$ representing the connections between agents $i$ and $j$. The neighbourhood set $N_i$ of node $i$ is the set of indices $j$ such that $\{i, j\}$ is an edge. We assume for simplicity that the graph is undirected, meaning that $\{i, j\}$ and $\{j, i\}$ represent the same edge.

The purpose of Kalman filtering is to provide an accurate estimate $\hat{x}_k$ of the state $x_k$ using the measurement $\tilde{y}_k$ and the known information about the system and noise. The standard discrete-time linear Kalman filter yields the estimate $\hat{x}_{k+1}$ and is given by

$$\hat{x}_{k+1} = A (\hat{x}_k + K_k [\tilde{y}_k - C \hat{x}_k])$$

$$K_k = P_k C^T \left[ CP_k C^T + R_k \right]^{-1}$$

$$P_{k+1} = A (I - K_k C) P_k A^T + \Sigma_k Q_k \Sigma_k^T,$$

where $K_k$ is the Kalman gain and $P_k$ is the error covariance matrix. The problem for high-dimensional systems is the inverse when computing $K_k$. In this paper, we assume that the sensor noise is uncorrelated, and so $R_k$ is a diagonal positive-definite matrix. This leaves the term $CP_k C^T$; if this term is diagonal, then the matrix inverse becomes a series of $n$ scalar divisions. We propose to replace $P_k$ with a diagonal estimate $\hat{P}_k$, and one can also assume that $C$ is a matrix that measures individual states of the system without redundancy. Therefore, a system with $1 \leq p \leq n$ outputs will have a measurement matrix of the form $C = [\text{diag}(C(i))]_p \mathbf{O}_{n-p}$. This assumption is common for networked systems where one measures the state of individual nodes of the network, see for example Chapman and Mesbahi (2015). In the next few sections, we will formalize these assertions in the context of our algorithm.

3. GENERAL KALMAN FILTERING

In this section, we extend the gradient descent estimate for the error covariance proposed by Sutton (1992). In §3.1, we discuss the linear Kalman filter, and then extend the error covariance estimate to an accelerated gradient descent in §3.2. The accelerated gradient descent benefits from a clever adaptive learning rate, which is discussed in §3.3. The final algorithm combining all these methods is summarized in Algorithm 4.

3.1 Gradient Descent for the Error Covariance Update: Linear Case

Gradient descent is an iterative algorithm that seeks to find the local minimum of a function $f$ by stepping in the direction of the largest gradient:

$$\beta_{k+1} = \beta_k - \mu \nabla f(\beta_k),$$

where $\mu$ is the learning rate. For the Kalman filter, we seek to find a diagonal matrix estimate $\hat{P}_k$ of $P_k$. We do this by assuming $\hat{P}_k$ is of the form $\hat{P}_k = \text{diag}(\hat{P}_k(i)) = \text{diag}(e^{\hat{P}_k})$, where $\hat{P}_k$ is a parameter undergoing gradient descent attempting to minimize the norm of the error $\delta_k = \tilde{y}_k - C \hat{x}_k$:

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \frac{\partial (\delta_k^T \delta_k)}{\partial \beta_k^{(i)}}.$$  

Computing the gradient of $\delta_k^T \delta_k$ as outlined in Appendix A yields the following gradient descent equations for $\hat{P}$:

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \frac{\partial (\delta_k^T \delta_k)}{\partial \beta_k^{(i)}}.$$  

The form of $\hat{P}^+$ guarantees that it remains positive-definite, which is required to preserve convergence of the Kalman filter.

3.2 Nesterov-Accelerated Methods

Nesterov acceleration is a method used to increase the convergence rate of gradient descent [Nesterov (1983)]. Although Nesterov-accelerated gradient descent converges in fewer timesteps, it does so by sacrificing monotonicity: the gradient descent trajectory will tend to oscillate as it converges towards the estimate.

In order to implement Nesterov acceleration, one must see that the quantity $h_k(i)$ is a function of $K_{k-1}$, which in turn is a function of $\beta_k^{(i)}$:

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \nabla f(\beta_k^{(i)})$$

$$h_k^{(i)} = h_k^{(i)} \left( 1 - k_k^{(i)} C^{(i)} \right) + \left( k_k^{(i)} - k_k^{(i)} C^{(i)} \right)^T \delta_k.$$  

Hence, we can get the form of the Nesterov-accelerated gradient descent by introducing the quantity $\alpha_k$:

$$\beta_{k+1}^{(i)} = \alpha_k^{(i)} - \frac{1}{2} \mu \nabla f(\beta_k^{(i)})$$

$$\alpha_k^{(i)} = \alpha_k^{(i)} + \mu \delta_k^T C^{(i)} h_k^{(i)} \left( K_{k-1} \left( \beta_k^{(i)} \right) \right).$$

where $\alpha_k^{(i)}$ is given by

$$\alpha_k^{(i)} = \beta_k^{(i)} + \frac{k - 1}{k + 2} \left( \beta_k^{(i)} - \beta_{k-1}^{(i)} \right).$$

Therefore, the Nesterov-accelerated estimate $\hat{P}$ replaces the covariance update with the following equations:

$$\hat{P} = \text{diag} \left( e^{\hat{P}_k^{(i)}} \right), \quad k_k^{(i)} = \exp \left( \alpha_k^{(i)} \right) D_k^{-1} C^{(i)}$$

$$\beta_{k+1}^{(i)} = \alpha_k^{(i)} + \mu \delta_k^T C^{(i)} h_k^{(i)} \left( K_{k-1} \left( \alpha_k^{(i)} \right) \right),$$

where $h_k$ is as in Equation (1), but using this new expression for $k_k^{(i)}$ in Equation (2).

3.3 Adaptive Learning Rate Methods

Using a constant learning rate $\mu$ can lead to suboptimal performance, and therefore it is prudent to adapt $\mu$ on-the-fly as the gradient descent is run. A well-known method for adapting the learning rate is given by Barzilai and Borwein Barzilai and Borwein (1988). Suppose we have the gradient descent with learning rate $\mu$:

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \nabla f(\beta_k^{(i)}).$$

$$\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \nabla f(\beta_k^{(i)}).$$
دریافت فوری
متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات