ESTIMATION AND DYNAMIC UPDATING OF TIME-VARYING SIGNALS WITH SPARSE VARIATIONS

M. Salman Asif, Adam Charles, Justin Romberg, and Christopher Rozell

School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA, USA

ABSTRACT

This paper presents an algorithm for an $\ell_1$-regularized Kalman filter. Given observations of a discrete-time linear dynamical system with sparse errors in the state evolution, we estimate the state sequence by solving an optimization algorithm that balances fidelity to the measurements (measured by the standard $\ell_2$ norm) against the sparsity of the innovations (measured using the $\ell_1$ norm). We also derive an efficient algorithm for updating the estimate as the system evolves. This dynamic updating algorithm uses a homotopy scheme that tracks the solution as new measurements are slowly worked into the system and old measurements are slowly removed. The effective cost of adding new measurements is a number of low-rank updates to the solution of a linear system of equations that is roughly proportional to the joint sparsity of all the innovations in the time interval of interest.

1. INTRODUCTION

In this paper we consider the classical problem of estimating a time-varying vector described by the discrete-time linear dynamical system

\[ y_i = A_i x_i + e_i, \]
\[ x_{i+1} = F_i x_i + w_i. \]

At each time step $i = 1, 2, \ldots$, we observe $M$ measurements $y_i \in \mathbb{R}^M$ of the unknown vectors $x_i \in \mathbb{R}^N$ through the system $A_i$; these measurements are corrupted by a noise vector $e_i$. The $N \times N$ matrix $F_i$ models the expected motion of $x_i$ to $x_{i+1}$; as the motion may drift from this expected behavior, we allow for an error $w_i$ to be incorporated at each step. We will assume throughout that the motion matrices $F_i$ are well-conditioned enough that we may confidently calculate $F_i^{-1}$. No such assumption is made for the $A_i$, which we will generally consider as underdetermined ($M < N$).

The equations in (1) are solved in the least-squares sense by the Kalman filter [1, 2]. Given the sequence of measurements $y_1, \ldots, y_k$, and starting from an initial estimate $\hat{x}_0$ of the vector at time $i = 0$, we can estimate $x_1, \ldots, x_k$ by solving the following optimization program

\[
\text{minimize}_{x_1, \ldots, x_k} \sum_{i=1}^{k} \tau_i \|x_i - F_{i-1} x_{i-1}\|^2 + \sum_{i=1}^{k} \lambda_i \|A_i x_i - y_i\|^2. \tag{2}
\]

The $\tau_i$ and $\lambda_i$ above are regularization parameters that we tune based on our relative confidence in the measurement and prediction errors — if the error vectors $e_i$ and $w_i$ are independent and identically distributed Gaussian noise vectors, then optimal choices of these parameters are based on the variances of the entries in this noise vector.

The Kalman filter owes its status as one of the pillars of statistical signal processing to the fact that the solution to (2) can be computed recursively. Say we have solved (2) after $k$ time steps, and denote the estimate of $x_k$ at this point as $\hat{x}_{k|k}$. Then given a new set of measurements $y_{k+1}$, we can efficiently update the estimate $\hat{x}_{k+1|k+1}$ of the new current state given only the previous estimate $\hat{x}_{k|k}$. If $A_{k+1}$ has $M$ rows, then the cost of this low-rank update is essentially the same as $M$ matrix-vector multiplies. Estimates for previous values of $x_i$ for $i < k + 1$ can be updated by working backwards (“smoothing”), with each step requiring a similar low-rank computation.

In this paper, we show how $\ell_1$ regularization can be incorporated into the Kalman filtering framework. The motivation for this is that there are many applications where we expect the innovations in the vector $x_i$ to be sparse. Our model replaces the second equation in (1) with

\[ x_{i+1} = F_i x_i + s_i, \]

where $s_i$ has only a few significant entries at unknown locations which are possibly different at each time instance. In place of the optimization program (2), we solve

\[
\text{minimize}_{x_1, \ldots, x_k} \sum_{i=1}^{k} \tau_i \|x_i - F_{i-1} x_{i-1}\|^2 + \sum_{i=1}^{k} \lambda_i \|A_i x_i - y_i\|^2, \tag{3}
\]

and show how the solution to this program can be efficiently updated as $k$ increases. Our algorithm maintains the optimal solution over a sliding window of time instances, and yields the jointly optimal solution given the measurements inside the frame and the estimate from the last frame out, which in this case is the initial estimate $\hat{x}_0$.

There have been several previous works with the same broad goal of incorporating discrete dynamics into sparse approximation, see in particular [3] and [4]. The formulation we give in this paper (see Section 2) varies slightly from those in these references, but our most important contribution is an efficient updating scheme for moving from one solution to the next as more observations are taken. This updating scheme, while not quite as simple as in the least-squares formulation, has the same recursive spirit as the original Kalman filter, and similarly reduces to a series of low-rank updates.

The paper is organized as follows. In section 2 we discuss $\ell_1$-regularized inverse problems and the formulation of our proposed dynamic recovery scheme. In section 3 we present a homotopy

Note that we are weighting all of the measurements the same, leaving out the $\lambda_i$ (although this is an easy extension.)
based dynamic updating algorithm for our proposed scheme. In section 4 we present some experimental results to demonstrate the performance of our scheme in estimation and tracking in the presence of sparse variations and study the performance gains from the dynamic update.

2. SPARSE VARIATION ESTIMATION

We start with a brief discussion of \( \ell_1 \) regularization for static inverse problems. Suppose we have observed \( y = Ax_0 + e \), and suppose that we have knowledge that the underlying vector \( x_0 \) is sparse. One way to take advantage of this sparsity when estimating \( x_0 \) from \( y \) is by searching for the vector with smallest \( \ell_1 \) norm that provides a reasonable explanation for the measurements. Given \( y \), we solve the following basis pursuit denoising (BPDN) [5] problem

\[
\text{minimize} \quad \tau \|x\|_1 + \frac{1}{2} \|Ax - y\|_2^2. \quad (4)
\]

For measurement matrices that obey certain incoherence conditions, (4) comes with a number of performance guarantees [6–10]. In particular if \( x_0 \) is sparse enough and there is no noise, (4) recovers \( x \) exactly as \( \tau \to 0 \) even though \( A \) is underdetermined, and the recovery is stable in the presence of noise with an appropriate choice of \( \tau \).

The optimization program (4) does not have a closed-form solution. It is typically solved using some type of iterative descent algorithm. Recently, recursive schemes based on homotopy have been proposed to update the solution to (4) when new measurements have been observed [11–13] (i.e. entries are added to \( y \) and rows are added to \( A \)) or when \( y \) is replaced with a completely new set of measurements of a closely related signal while keeping \( A \) constant [12, 14]. These algorithms are derived by working the new measurements into (or out of) the optimization program gradually, and tracking how the solution changes. The problems are formulated such that the movement from one solution to the next can be broken down into a series of linear steps, with each link traversed using a low-rank update.

Our program (3) might be considered the dynamic version of (4). We start by reformulating the \( \ell_1 \) version of the Kalman filter in terms of the sparse innovations \( s_i \); we will explicitly estimate the \( s_i \) and then turn these into estimates for the \( x_i \) using the recursive equation \( \hat{x}_i = F_{i-1}x_{i-1} + \hat{s}_i \) (recall that we are working from a fixed initial estimate \( \hat{x}_0 \)). The sparse innovations can be written as

\[
\begin{bmatrix}
    s_1 \\
    s_2 \\
    \vdots \\
    s_{k-2} \\
    s_{k-1}
\end{bmatrix} =
\begin{bmatrix}
    I & 0 & \cdots & 0 & 0 \\
    -F_1 & I & \cdots & 0 & 0 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    0 & 0 & \cdots & -F_{k-2} & I \\
    0 & 0 & \cdots & 0 & -F_{k-1}
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_{k-2} \\
    x_{k-1}
\end{bmatrix}
- \begin{bmatrix}
    F_0x_0 \\
    0 \\
    \vdots \\
    0 \\
    0
\end{bmatrix}
\equiv s = Fx - u. \quad (5)
\]

Note that

\[
F^{-1} =
\begin{bmatrix}
    I & 0 & 0 & \cdots & 0 & 0 \\
    F_1 & I & 0 & \cdots & 0 & 0 \\
    F_2F_1 & F_2 & I & \cdots & 0 & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    F_{k-1}F_{k-2}F_{k-3} \cdots F_k & F_{k-1}F_{k-2}F_{k-3} \cdots F_k & F_{k-1}F_{k-2}F_{k-3} \cdots F_k & I
\end{bmatrix}
\]

where \( F_{k-1} = F_kF_{k-1} \cdots F_i \). Since \( x \) can be written as \( x = F^{-1}(s + u) \), we have the following equalities

\[
Ax = AF^{-1}s + AF^{-1}u,
\]

\[
\sum_{i=1}^{k}\|A_i x_i - y_i\|_2^2 = \|Ax - y\|_2^2 = \|AF^{-1}s - (y - AF^{-1}u)\|_2^2,
\]

where

\[
A =
\begin{bmatrix}
    A_1 & 0 & \cdots & 0 \\
    0 & A_2 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & A_k
\end{bmatrix}
\quad \text{and} \quad y =
\begin{bmatrix}
    y_1 \\
    y_2 \\
    \vdots \\
    y_k
\end{bmatrix}
\]

Therefore, we can write the optimization problem in (3) with variables \( x_1, \ldots, x_k \), in the form of following optimization problem with variables \( s_1, \ldots, s_k \),

\[
\text{minimize} \quad \sum_{i=1}^{k} \|s_i\|_1 + \frac{1}{2} \|AF^{-1}s - \tilde{y}\|_2^2, \quad (6)
\]

where \( \tilde{y} = y - AF^{-1}u \), and \( u \) is as in (5). The optimization problem in (6) is a weighted BPDN problem. We can solve this problem to get the sparse solution \( \hat{s} \), which in turn gives us an estimate for \( x \).

Note that the size of the problem in (6) increases as we add new measurements in the working set \( A \). We will work on a sliding window, considering a fixed number \( P \) of time instances jointly. The desired length and instances of working system can be chosen by appropriately selecting the indices in (3) and (6). For example, suppose we want to keep the size of working system fixed at \( PM \times PN \), and we have solved (6) at time \( P \). Now we want to add \( M \) new measurement to the system and remove the oldest \( M \) measurements. Therefore, we solve (6) with \( i = 2, \ldots, P + 1 \). The main question here is whether we can quickly update the solution for (6) with index set \( i = 2, \ldots, P + 1 \) given a previously calculated solution for \( i = 1, \ldots, P \); instead of requiring to solve it from scratch. Fortunately, the answer to this question is affirmative, and in the next section we devise a dynamic updating scheme to perform this task.

3. DYNAMIC UPDATING ALGORITHM

In this section we present an algorithm to dynamically update the solution of (3) and (6) as measurements are added or removed. To perform this task, we formulate a homotopy path between the original and updated solutions, similar to the ones used in [12], and demonstrate how this path can be traversed.

Suppose we have solved (6) for time indices \( i = 1, \ldots, P \), and call the estimate for the innovations \( \hat{s} \) and the corresponding state estimate \( \tilde{x} = F^{-1}(\hat{s} + u) \) (which solves (3)). We break down the shift to the new time interval of interest to \( i = 2, \ldots, P + 1 \) into a number of small steps, each with a predictable effect on the solution. We first remove the \( M \) rows corresponding to \( A_i \) and \( N \) columns corresponding to the first innovation \( s_1 \) from \( AF^{-1} \). Following this, \( N \) new columns corresponding to the new innovation \( s_{P+1} \) are appended to \( AF^{-1} \). Finally, a new set of \( M \) rows are added corresponding to \( AP_{P+1} \). Below we describe how to track the solution \( \hat{s} \) to our \( \ell_1 \) minimization program as the system undergoes these transformations. These paths between solutions will break down into a series of linear segments, each of which can be traversed very quickly. The computational cost of the algorithm scales with the number of segments in the path, which in turn is determined by the number of active elements in the set of innovations. The update will be efficient.
if state solution changes only at small number of locations over the course of its evolution.

Denote the current state solution as \( \hat{x} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_P] \).

Our first claim is that if we remove \( A_1 \) and \( x_1 \) from the system and solve (3) for the index set \( i = 2, \ldots, P \), using \( F_i \hat{x}_i \) as the initial estimate of \( x_2 \), then the solution remains as \( [\hat{x}_2, \ldots, \hat{x}_P] \). This property can be verified by looking at the following optimality conditions of (3), which hold for all \( i = 1, \ldots, P \):

\[
A_i^T(A_i x_i - y_i) + \sum_{j=1+1}^{P-1} F_j^T \cdots F_{i-1}^T A_i^T(A_i x_j - y_j) = -u_i,
\]

\[
u_i(\hat{x}) = \begin{cases} 1 & \text{if } |x_i - F_{i-1} x_{i-1}|(k) \neq 0 \\ 0 & \text{if } |x_i - F_{i-1} x_{i-1}|(k) = 0. \end{cases}
\] (7)

Note that the optimality conditions have a recursive relationship: if the solution \( \hat{x}_2 \) is not disturbed by removing \( x_1 \) from the system, then all the subsequent solutions \( \hat{x}_3, \ldots, \hat{x}_P \) must remain fixed. Since (7) for \( i = 2 \) is already satisfied with \( x_1 = \hat{x}_1 \) and \( x_2 = \hat{x}_2 \), our solution does not change if we remove \( x_1 \) from the system and use its estimate to form the estimate of \( x_2 \). It is clear that this argument extends beyond the first set of measurements. We can remove measurements up to any index \( 1 \leq j < P \) and use \( F_j \hat{x}_j \) as the initial estimate for \( x_{j+1} \) without changing the remaining solutions.

Let us now look at what happens when we add a new signal instance \( x_{P+1} \) into the system. Note that it does not affect the old solution either. This can be seen by looking at (3), where increasing the end index for prediction error term from \( P \) to \( P + 1 \) adds \( x_{P+1} \) into the working system. Before adding any measurements for \( x_{P+1} \), its optimal solution will be \( \hat{x}_{P+1} = F_P \hat{x}_P \), which implies that initial value for \( s_{P+1} = 0 \).

With the stale measurements removed and our innovation initialized to \( s_{P+1} = 0 \), we are ready for the main task of updating all of the innovation estimates \( \hat{s}_2, \ldots, \hat{s}_{P+1} \) as we introduce the new measurements \( y_{P+1} \) into the system. We do this update using a homotopy continuation. The basic idea in the homotopy continuation is to transform a given optimization problem into a related problem, for which the solution is either already available or can be computed easily. The continuation from the transformed problem to the original problem can then be done in a series of low-complexity updates.

Let us assume that \( A, F^{-1}, x, s \) and \( y \) denote the system parameters corresponding to the new time window \( i = 2, \ldots, P + 1 \). We compute the new \( \tilde{y} = y - AF^{-1}u \), where \( u \) is in turn computed from \( F_i \hat{x}_i \) in (5), and set \( \tilde{u}_1 = [\tilde{s}_2, \ldots, \tilde{s}_{P+1}] \). Denote first \( (P - 1)M \) rows of the new system matrix as \( U \), the last \( M \) rows of the matrix as \( B \), the first \( (P - 1)M \) elements of \( \tilde{y} \) as \( v \), and the last \( M \) elements as \( w \). To make the exposition as simple as possible, we assume that \( \tau_i = \tau \) for all \( i \). We can now move from the old solution to the new by slowly increasing \( \epsilon \) in the following homotopy formulation:

\[
\min_{s} \|y - AB^T x\|_1 + \frac{1}{2} \|Us - \tilde{v}\|_2^2 + \frac{1}{2} \|Bs - \tilde{w} - (1 - \epsilon)w_{old}\|_2^2.
\] (8)

where \( w_{old} = BS \). At \( \epsilon = 0 \), \( s_1 \) is the solution of (8). As we increase \( \epsilon \) towards 1, the solution of (8) approaches the solution of (6) for the index set \( i = 2, \ldots, P + 1 \).

We traverse the solution path parameterized by \( \epsilon \) in (8) using the same basic principles detailed in [12]. The solution \( s^* \) of (8) at any given value of \( \epsilon \) must obey the following optimality conditions

\[
U_i^T(Us^* - \tilde{v}) + B_i^T( Bs^* - \tilde{w} - (1 - \epsilon)w_{old}) = -\tau \xi
\] (9)

\[
\|U_i^T(Us^* - \tilde{v}) + B_i^T( Bs^* - \tilde{w} - (1 - \epsilon)w_{old})\|_\infty < \tau
\] (10)

where \( \Gamma \) denotes the support of \( s^* \), \( B_i \) is the matrix formed from the columns of \( B \) indexed by \( i \), and \( \xi \) is the sign sequence of \( s^* \) on \( \Gamma \). Notice that if \( s \) follows a piecewise linear path as \( \epsilon \) varies, this path changes direction when either an element of the solution shrink to zero, or the inequality constraint (9) is violated (that is, the support of the solution changes). Suppose we are at a solution \( s^* \) to (8) with support \( \Gamma \) and signs \( z \) at some given value of \( \epsilon = \epsilon_0 \) between zero and one. As we increase \( \epsilon \) by an infinitesimal amount to \( \epsilon + \theta \), the solution moves to \( s^{k+1} = s^* + \theta ds \), where

\[
\partial s = (U_i^T U + B_i^T B)^{−1} B_i^T (\tilde{w} - w_{old}).
\] (11)

Moving in the direction of \( \partial s \) by increasing the step size \( \theta \), we eventually hit a critical point where either one of the entries in \( s^k \) shrinks to zero or one of the constraints in (10) becomes active (equal to \( \tau \)). The smallest amount we can move \( \epsilon \) so that the former is true is simply

\[
\theta^- = \min_{j \in \Gamma} (\partial s(\tilde{v})) + \min(\epsilon_j - \epsilon_0)\epsilon_j = \min(\epsilon_j - \epsilon_0)\epsilon_j.
\] (12)

where \( \min(\epsilon_j) \) denotes that the minimum is taken over positive arguments only. For the latter, set

\[
p_k = U_i^T (Us^k - \tilde{v}) + B_i^T (Bs^k - \tilde{w} - (1 - \epsilon_k)w_{old})
\] (13a)

\[
d_k = (U_i^T U + B_i^T B)ds - B_i^T (\tilde{w} - w_{old}).
\] (13b)

We are now looking for the smallest stepsize \( \Delta \) so that \( p_k(j) + \Delta \epsilon \cdot d_k(j) = \pm \tau \) for some \( j \in \Gamma \). This is given by

\[
\theta^+ = \min_{j \in \Gamma} \left( \tau - p_k(j) + \tau - p_k(j) \right)
\] (14)

The stepsize to the next critical point is

\[
\theta = \min(\theta^+, \theta^-).
\] (15)

With the direction \( \partial s \) and stepsize \( \theta \) chosen, the next critical value of \( \epsilon \) and the solution at that point become

\[
\epsilon_{k+1} = \epsilon_k + \theta, \quad s^{k+1} = s^k + \theta ds
\]

The support for new solution \( s^{k+1} \) differs from \( \Gamma \) by one element. Let \( \gamma^- \) be the index for the minimizer in (12) and \( \gamma^+ \) be the index for the minimizer in (14). If we chose \( \theta^- \) in (15), then we remove \( \gamma^- \) from the support \( \Gamma \) and the sign sequence \( z \). If we chose \( \theta^+ \) in (15), then we add \( \gamma^+ \) to the support, and add the corresponding sign to \( z \). This procedure is repeated until \( \epsilon = 1 \).

The main computational cost at every homotopy step comes from solving a \( |\Gamma| \times |\Gamma| \) system of equations to compute the direction in (11), and two matrix-vector multiplications to compute the \( d_k \) for the stepsize. Since the support changes by a single element at every homotopy step, the update direction can be computed using rank-1 update methods [15]. Furthermore, the lower block-triangular structure of \( U \) and \( B \) can be exploited to further expedite the processing and reduce the storage requirements.

4. EXPERIMENTS

In this section we present some initial experimental results to demonstrate the performance of our proposed scheme. We first analyze the performance of (3) for estimation and tracking of signals in the presence of sparse variations. Next we analyze the advantage of our proposed dynamic updating scheme over the standard BPDN homotopy solver.
In our experimental setup we follow the dynamical system model in (1) and (1'), taking $F_i$ as the identity for all $i$ and every $A_i$ as a random matrix whose entries are independently chosen from the standard Normal distribution. Our initial signal $x_0$ can be an arbitrary vector. We generate every sparse variation vector $s_i$ independently by selecting $K$ locations at random and assigning them amplitudes following Normal distribution. We synthesize the measurements $y_i$ and present them as input into our algorithm along with the initial signal estimate $\hat{x}_0$ and regularization parameter $\tau$.

In order to evaluate the estimation and tracking performance of (3), we solved (6) for the same dynamical system using different values of $P$ and innovation sparsity $K$. We initialized $x_0$ as a vector whose all entries are normally distributed and used zero vector as $\hat{x}_0$. The recovery results for a system with $N = 128$ and $M = 64$ are plotted in Figure 1, with noiseless measurements (left plot) and in the presence of Gaussian noise at 30dB (right plot). The value at each point is recorded by averaging the results of 10 simulations, each run over 500 time instances of the system. The performance metric is the signal to reconstruction error ratio (SER) defined as

$$\text{SER} = 20 \log_{10} \frac{\|x\|_2}{\|x - \hat{x}\|_2},$$

where $x$ denotes the original signal and $\hat{x}$ denotes the reconstructed signal. Note that the signal estimation improves considerably as we increase $P$. The superior performance with larger values of $P$ is a reflection of better signal estimation and tracking of the sparse changes in the signal dynamics.

Next we compare the cost of solving (6) from scratch using standard BPDN homotopy [16, 17] with our proposed dynamic updating algorithm. We compare the number of homotopy steps taken by each method for solving (6), using the experimental setup mentioned above, except here we initialize $x_0$ as a $K$-sparse vector. We have tabulated the average number of homotopy steps taken by both solvers for different values of $P$ in Table 1. For small values of $P$ both methods take approximately the same number of homotopy steps to solve (6). However, for larger values of $P$, the number of dynamic update steps is significantly smaller than the number of BPDN homotopy steps.

5. CONCLUSION

We have presented an $\ell_1$-regularized algorithm for estimating time-varying signals with sparse prediction errors in the settings of a linear dynamical system. While a globally optimal solution would include all previous states, the computational complexity of solving such system would be overwhelming. We presented a computationally feasible algorithm which works over a sliding window of time instances, and can dynamically update the solution using homotopy updating scheme. Utilizing this algorithm, additional dynamical signal models that are not covered by the conventional Kalman filter, can be estimated in a precise and efficient manner.

### 6. REFERENCES


