Dynamic Reconstruction: Least-squares and $\ell_1$

Justin Romberg, Georgia Tech ECE
CCNS Summer School, SAMS, RTP, NC
July 27, 2015
Goal: a dynamical framework for sparse recovery

\textit{Given }y \textit{ and } \Phi \textit{, solve}

\[
\min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y\|_2^2
\]
Goal: a dynamical framework for sparse recovery

We want to move from:

\[ \text{Given } y \text{ and } \Phi, \text{ solve } \]
\[ \min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y\|_2^2 \]

to

\[ y(t) \left\{ \text{min } \ell_1 \right\} \Phi(t) \rightarrow \hat{x}(t) \]
Classical: Recursive least-squares

- System model:
  \[ y = \Phi x \]
  - \( \Phi \) has full column rank
  - \( x \) is arbitrary

- Least-squares estimate:
  \[
  \min ||y - \Phi x||^2_2 \implies \hat{x} = (\Phi^T \Phi)^{-1} \Phi^T y
  \]
Classical: Recursive least-squares

- Sequential measurement:
  \[
  \begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} \Phi \\ \phi^T \end{bmatrix} x
  \]

- Recursive updates
  
  \[ w \cdot y \cdot w = \cdot \cdot \cdot \]
  
  \[ x = \cdot \cdot \cdot \]

  Compute new estimate using rank-1 update:
  \[
  \hat{x}_1 = (\Phi^T \Phi + \phi \phi^T)^{-1}(\Phi^T y + \phi \cdot w)
  = \hat{x}_0 + K_1(w - \phi^T x_0)
  \]

  where
  \[
  K_1 = (\Phi^T \Phi)^{-1} \phi (1 + \phi^T (\Phi^T \Phi)^{-1} \phi)^{-1}
  \]

  With the previous inverse in hand, the update has the cost of a few matrix-vector multiplies.
Classical: The Kalman filter

- Linear dynamical system for state evolution and measurement:
  \[ y_t = \Phi_t x_t + e_t \]
  \[ x_{t+1} = F_t x_t + d_t \]

\[
\begin{bmatrix}
I & 0 & 0 & 0 & \cdots \\
\Phi_1 & 0 & 0 & 0 & \cdots \\
-F_1 & I & 0 & 0 & \cdots \\
0 & \Phi_2 & 0 & 0 & \cdots \\
0 & -F_2 & I & 0 & \cdots \\
0 & 0 & \Phi_3 & 0 & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots
\end{bmatrix}
= 
\begin{bmatrix}
F_0 x_0 \\
y_1 \\
y_2 \\
y_3 \\
\vdots
\end{bmatrix}
\]

- As time marches on, we add both rows and columns.
- Least-squares problem:
  \[
  \min_{x_1, x_2, \ldots} \sum_t \left( \sigma_t \| \Phi_t x_t - y_t \|_2^2 + \lambda_t \| x_t - F_{t-1} x_{t-1} \|_2^2 \right)
  \]
Say we take three measurements of a patient’s pulse; each (scalar) measurement has form

\[ y = x + \text{noise} \]

\( y = \) measured pulse, \( x = \) true pulse

After each measurement, what is our best guess of the pulse \( x \)?

(“best” = least-squares optimal)
Static least-squares vs. Kalman filter example

\[ y_i = x_* + \text{noise} \]

\[ y = \text{measured pulse}, \; x = \text{true pulse} \]

Systems of equations:

\[
\begin{bmatrix}
y_0 \\ y_1
\end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x_* + \begin{bmatrix} e_0 \\ e_1 \end{bmatrix}, \quad \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} x_* + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \end{bmatrix}
\]

Solve each with least-squares:

\[
\hat{x}_i = \left( A_i^T A_i \right)^{-1} A_i^T \begin{bmatrix} y_0 \\ \vdots \\ y_i \end{bmatrix}
\]
Static least-squares vs. Kalman filter example

\[ y_i = x_* + \text{noise} \]

\( y = \text{measured pulse}, \ x = \text{true pulse} \)

Systems of equations:

\[
\begin{bmatrix}
  y_0 \\
  y_1
\end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x_* + \begin{bmatrix} e_0 \\ e_1 \end{bmatrix},
\]

\[
\begin{bmatrix}
  y_0 \\
  y_1 \\
  y_2
\end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} x_* + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \end{bmatrix}
\]

\[ \hat{x}_0 = y_0, \quad \hat{x}_1 = \frac{y_0 + y_1}{2}, \quad \hat{x}_2 = \frac{y_0 + y_1 + y_2}{3} \]
Static least-squares vs. Kalman filter example

With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\[ y_i = \text{measured pulse at time } i, \quad x_i = \text{true pulse at time } i \]
With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i \) = measured pulse at time \( i \), \( x_i \) = true pulse at time \( i \)

System of equations, time \( i = 0 \):

\[ y_0 = x_0 + e_0 \]
Static least-squares vs. Kalman filter example

With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i \) = measured pulse at time \( i \), \( x_i \) = true pulse at time \( i \)

System of equations, time \( i = 1 \):

\[
\begin{bmatrix}
  y_0 \\
  0 \\
  y_1
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 \\
  -1 & 1 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1
\end{bmatrix}
+ \begin{bmatrix}
  e_0 \\
  e_{01} \\
  e_1
\end{bmatrix}
\]
With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i \) = measured pulse at time \( i \), \( x_i \) = true pulse at time \( i \)

System of equations, time \( i = 2 \):

\[
\begin{bmatrix}
  y_0 \\
  y_1 \\
  y_2
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 \\
  -1 & 1 & 0 \\
  0 & 1 & 0 \\
  0 & -1 & 1 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1 \\
  x_2
\end{bmatrix}
\]
With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i \) = measured pulse at time \( i \), \( x_i \) = true pulse at time \( i \)

System of equations, time \( i = 0 \):

\[ y_0 = x_0 + e_0 \]

LS solution:

\[ \hat{x}_{0|0} = y_0 \]
Static least-squares vs. Kalman filter example

With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i = \) measured pulse at time \( i \), \( x_i = \) true pulse at time \( i \)

System of equations, time \( i = 1 \):

\[
\begin{bmatrix}
  y_0 \\
  0 \\
  y_1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 \\
  -1 & 1 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1
\end{bmatrix} +
\begin{bmatrix}
  e_0 \\
  e_{01} \\
  e_1
\end{bmatrix}
\]

LS solution:

\[
\begin{bmatrix}
  \hat{x}_0 | 1 \\
  \hat{x}_1 | 1
\end{bmatrix} =
\left(\begin{bmatrix}
  1 & -1 & 0 \\
  0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
  1 & 0 \\
  -1 & 1 \\
  0 & 1
\end{bmatrix}\right)^{-1}
\begin{bmatrix}
  1 & -1 & 0 \\
  0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
  y_0 \\
  0 \\
  y_1
\end{bmatrix}
\]
With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i \) = measured pulse at time \( i \), \( x_i \) = true pulse at time \( i \)

System of equations, time \( i = 1 \):

\[
\begin{bmatrix}
  y_0 \\
  0 \\
  y_1 \\
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 \\
  -1 & 1 \\
  0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1 \\
\end{bmatrix} +
\begin{bmatrix}
  e_0 \\
  e_{01} \\
  e_1 \\
\end{bmatrix}
\]

LS solution:

\[
\begin{bmatrix}
  \hat{x}_{0|1} \\
  \hat{x}_{1|1} \\
\end{bmatrix} = \left(\begin{bmatrix}
  2 & -1 \\
  -1 & 2 \\
\end{bmatrix}\right)^{-1}
\begin{bmatrix}
  y_0 \\
  y_1 \\
\end{bmatrix}
\]

Static least-squares vs. Kalman filter example
With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i \) = measured pulse at time \( i \), \( x_i \) = true pulse at time \( i \)

System of equations, time \( i = 1 \):

\[
\begin{bmatrix}
  y_0 \\
  0 \\
  y_1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 \\
  -1 & 1 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1
\end{bmatrix} +
\begin{bmatrix}
  e_0 \\
  e_{01} \\
  e_1
\end{bmatrix}
\]

LS solution:

\[
\begin{bmatrix}
  \hat{x}_0|_1 \\
  \hat{x}_1|_1
\end{bmatrix} =
\begin{bmatrix}
  2/3 & 1/3 \\
  1/3 & 2/3
\end{bmatrix}
\begin{bmatrix}
  y_0 \\
  y_1
\end{bmatrix}
\]
Static least-squares vs. Kalman filter example

With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\[ y_i = \text{measured pulse at time } i, \quad x_i = \text{true pulse at time } i \]

System of equations, time \( i = 1 \):

\[
\begin{bmatrix}
  y_0 \\
  0 \\
  y_1
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 \\
  -1 & 1 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1
\end{bmatrix}
+ \begin{bmatrix}
  e_0 \\
  e_{01} \\
  e_1
\end{bmatrix}
\]

LS solution:

\[
\begin{bmatrix}
  \hat{x}_{0|1} \\
  \hat{x}_{1|1}
\end{bmatrix}
= \begin{bmatrix}
  \frac{2y_0 + y_1}{3} \\
  \frac{y_0 + 2y_1}{3}
\end{bmatrix}
\]
Static least-squares vs. Kalman filter example

With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\[ y_i = \text{measured pulse at time } i, \; x_i = \text{true pulse at time } i \]

System of equations, time \( i = 2 \):

\[
\begin{bmatrix}
y_0 \\
y_1 \\
y_2
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 & 0 \\
-1 & 1 & 0 \\
0 & 1 & 0 \\
0 & -1 & 1 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2
\end{bmatrix}
\]

LS solution:

\[
\begin{bmatrix}
\hat{x}_{0|2} \\
\hat{x}_{1|2} \\
\hat{x}_{1|3}
\end{bmatrix}
= 
\left( \begin{bmatrix}
2 & -1 & 0 \\
-1 & 3 & -1 \\
0 & -1 & 2
\end{bmatrix} \right)^{-1}
\begin{bmatrix}
y_0 \\
y_1 \\
y_2
\end{bmatrix}
\]
Static least-squares vs. Kalman filter example

With the Kalman filter framework, the pulse can drift:

\[ y_i = x_i + \text{noise}, \quad x_{i+1} = x_i + \text{noise} \]

\( y_i = \) measured pulse at time \( i \), \( x_i = \) true pulse at time \( i \)

System of equations, time \( i = 2 \):

\[
\begin{bmatrix}
  y_0 \\
  y_1 \\
  y_2 \\
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 \\
  -1 & 1 & 0 \\
  0 & 1 & 0 \\
  0 & -1 & 1 \\
  0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
\end{bmatrix}
\]

LS solution:

\[
\begin{bmatrix}
  \hat{x}_0|2 \\
  \hat{x}_1|2 \\
  \hat{x}_1|3 \\
\end{bmatrix} =
\begin{bmatrix}
  \frac{5y_0 + 2y_1 + y_2}{8} \\
  \frac{y_0 + 2y_1 + y_2}{4} \\
  \frac{y_0 + 2y_1 + 5y_2}{8} \\
\end{bmatrix}
\]
Classical: The Kalman filter

- Linear dynamical system for state evolution and measurement:
  \[ y_t = \Phi_t x_t + e_t \]
  \[ x_{t+1} = F_t x_t + d_t \]

- Least-squares problem:
  \[
  \min_{x_1, x_2, \ldots} \sum_t \left( \sigma_t \| \Phi_t x_t - y_t \|_2^2 + \lambda_t \| x_t - F_{t-1} x_{t-1} \|_2^2 \right)
  \]

- Again, we can use low-rank updating to solve this recursively:
  \[ v_k = F_k \hat{x} \]
  \[ K_{k+1} = (F_k P_k F_k^T + I) \Phi_{k+1}^T (\Phi_{k+1} (F_k P_k F_k^T + I) \Phi_{k+1}^T + I)^{-1} \]
  \[ \hat{x}_{k+1|k+1} = v_k + K_{k+1} (y_{k+1} - \Phi_{k+1} v_k) \]
  \[ P_{k+1} = (I - K_{k+1} \Phi_{k+1})(F_k P_k F_k^T + I) \]
Streaming Least-Squares Reconstruction
Basic problem

Reconstruct a signal from \textit{generalized, local samples}.

Example to fix ideas: \textit{non-uniform samples}:

\[ y = \Phi x \]

Once discretized, this is a \textit{linear inverse problem}

Goal: reconstruct the signal “online” from streaming measurements
(Reconstruct \(\equiv\) produce equally spaced samples at appropriate rate)
Basic problem

Reconstruct a signal from *generalized, local samples*.

Example to fix ideas: *non-uniform samples*:

Once discretized, this is a *linear inverse problem*

\[ y = \Phi x \]

Goal: reconstruct the signal “online” from streaming measurements

Main results: local stability $\Rightarrow$ global stability and fast convergence
Signal model

Decompose the signal into frame bundles that are compactly supported in time:

\[ x(t) = \sum_{k \in \mathbb{Z}} \sum_{n=1}^{N} \alpha_{k,n} \psi_{k,n}(t) \]

The vector \( \alpha_k \in \mathbb{R}^N \) characterizes \( x(t) \) in a time window of length \( L \).

**Example:** Lapped Orthogonal Basis:

\[ \psi_{k,n}(t) = \sqrt{\frac{2}{L}} \cdot g \left( \frac{t - k}{L} \right) \cos \left[ \pi \left( n + \frac{1}{2} \right) \left( \frac{t - k}{L} \right) \right]. \]
Sample batch $k$ at locations $t_1, \ldots, t_M$

One batch overlaps frame bundles $k - 1$ and $k$

Single sample

$$x(t_m) = \sum_n \alpha_{k-1,n} \psi_{k-1,n}(t_m) + \sum_n \alpha_{k,n} \psi_{k,n}(t_m)$$

Collecting all samples into vector $y_k$, we can write

$$y_k = \begin{bmatrix} B_k & A_k \end{bmatrix} \begin{bmatrix} \alpha_{k-1} \\ \alpha_k \end{bmatrix}$$
Multiple batches

After collecting batches $k = 0, 1, \ldots, K$, we have the (possibly large) system

\[
\begin{bmatrix}
A_0 & 0 & \cdots & 0 \\
B_1 & A_1 & 0 & \cdots & 0 \\
0 & B_2 & A_2 & 0 & \cdots & 0 \\
0 & 0 & B_3 & A_3 & 0 & \cdots & 0 \\
0 & 0 & 0 & B_4 & A_4 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & B_K & A_K & \cdots & 0 \\
\end{bmatrix}
\begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_2 \\
\alpha_3 \\
\alpha_4 \\
\vdots \\
\alpha_K \\
\end{bmatrix}
=
\begin{bmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3 \\
y_4 \\
\vdots \\
y_K \\
\end{bmatrix}.
\]

More compactly,

\[
\Phi_K a_K = y_K.
\]
Least-squares

After collecting sample batches $y_k$ for $k = 0, 1, \ldots, K$, the least-squares estimate is

$$\hat{a}_K = (\Phi_K^T \Phi_K)^{-1} \Phi_K^T y_K.$$ 

The estimates for all previous bundles will vary with $K$; we write

$$\hat{a}_K = \begin{bmatrix}
\hat{\alpha}_{0|K} \\
\hat{\alpha}_{1|K} \\
\hat{\alpha}_{2|K} \\
\vdots \\
\hat{\alpha}_{K|K}
\end{bmatrix}.$$
The least-squares system is tri-diagonal,

\[
\Phi_K^T \Phi_K = \begin{bmatrix}
D_0 & E_0^T & 0 & \cdots & 0 \\
E_0 & D_1 & E_1^T & 0 & \cdots & 0 \\
0 & E_1 & D_2 & E_2^T & 0 & \cdots & 0 \\
0 & 0 & E_2 & D_3 & E_3^T & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & E_{K-2} & D_{K-1} & E_{K-1}^T & \cdots & 0 \\
0 & \cdots & 0 & E_{K-1} & D'_K
\end{bmatrix},
\]

where

\[
D_k = A_k^T A_k + B_{k+1}^T B_{k+1}, \quad k = 0, \ldots, K - 1 \\
D'_K = A_K^T A_K \\
E_k = B_{k+1}^T A_{k+1}, \quad k = 0, \ldots, K - 1.
\]
There is an easy LU factorization,

\[
\Phi_T K \Phi_K = \begin{bmatrix}
Q_0 & 0 & \cdots & 0 \\
E_0 & Q_1 & 0 & \cdots \\
0 & E_1 & Q_2 & \cdots \\
\vdots & \vdots & \ddots & \ddots \\
0 & \cdots & 0 & E_{K-1} & Q'_K
\end{bmatrix}
\begin{bmatrix}
I & U_0 & 0 \\
0 & I & U_1 & 0 \\
\vdots & \vdots & \ddots & \ddots \\
0 & 0 & \cdots & U_{K-1}
\end{bmatrix},
\]

where the \( Q_k \) and \( U_k \) can be computed \textit{recursively}.
Factorization

There is an easy LU factorization,

\[
\Phi^T_K \Phi_K = \begin{bmatrix}
Q_0 & 0 & \cdots & 0 \\
E_0 & Q_1 & 0 & \\
0 & E_1 & Q_2 & \cdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & E_{K-1} & Q'_K
\end{bmatrix}
\begin{bmatrix}
I & U_0 & 0 \\
0 & I & U_1 & 0 \\
\vdots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & I
\end{bmatrix},
\]

where the \( Q_k \) and \( U_k \) can be computed recursively

\[
Q_0 = D_0
\]

for \( k = 0, 2, \ldots, K - 1 \)

\[
U_{k-1} = Q_{k-1}^{-1} E_{k-1}^T
\]

\[
Q_k = D_k - E_{k-1} Q_{k-1}^{-1} E_{k-1}^T
\]

end

\[
Q'_K = D'_K - E_{K-1} U_{K-1}
\]
With estimates after $K$ batches in hand

$$\hat{\alpha}_0|_K, \hat{\alpha}_1|_K, \ldots, \hat{\alpha}_K|_K$$

we introduce a new batch of $M$ samples

$$y_{K+1} = A_{K+1}\alpha_{K+1} + B_{K+1}\alpha_K.$$ 

We can now update the factorization ($M \times M$ system solve), solve for $\alpha_{K+1}|_{K+1}$, then backtrack
Streaming least-squares

With estimates after $K$ batches in hand

$$\hat{\alpha}_{0|K}, \hat{\alpha}_{1|K}, \ldots, \hat{\alpha}_{K|K}$$

we introduce a new batch of $M$ samples

$$y_{K+1} = A_{K+1}\alpha_{K+1} + B_{K+1}\alpha_{K}.\quad$$

We can now *update* the factorization ($M \times M$ system solve), *solve* for $\alpha_{K+1|K+1}$, then *backtrack*

$$\nu_{K+1} = Q_{K+1}^{-1}(A_{K+1}^T y_{K+1} + B_{K+1}^T y_{K+1} - E_K \hat{\alpha}_{K|K})$$

$$\hat{\alpha}_{K+1|K+1} = \nu_{K+1}$$

for $k = K, K - 1, \ldots, 0$

$$\hat{\alpha}_{k|K+1} = \nu_k - U_k \hat{\alpha}_{k+1|K+1}$$

end
The stability of this procedure depends on the conditioning of the

\[ Q_k = D_k - E_{k-1} Q_{k-1}^{-1} E_{k-1}, \]

where

\[ D_k = A_k^T A_k + B_{k+1}^T B_{k+1}, \quad E_{k-1} = B_k^T A_k \]
Conditioning of the $Q_k$

The stability of this procedure depends on the conditioning of the

$$Q_k = D_k - E_{k-1}Q_{k-1}^{-1}E_{k-1},$$

where

$$D_k = A_k^T A_k + B_{k+1}^T B_{k+1}, \quad E_{k-1} = B_k^T A_k$$

If the local measurement matrices $D_k$ are well-conditioned,

$$1 - \delta \leq \lambda_{\min}(D_k) \leq \lambda_{\min}(D_k) \leq 1 + \delta, \quad \text{for all } k$$

and

$$\|E_k\| \leq \delta, \quad \text{for all } k,$$

with $\delta \leq 0.285$, then

$$1 - \delta_* \leq \lambda_{\min}(Q_k) \leq \lambda_{\min}(Q_k) \leq 1 + \delta_*, \quad \text{for all } k$$

for $\delta_* \leq 1.25 \delta$. 
Convergence of the corrections

Under the same conditions on $D_k, E_k$,

$$1 - \delta \leq \lambda_{\text{min}}(D_k) \leq \lambda_{\text{min}}(D_k) \leq 1 + \delta, \quad \|E_k\| \leq \delta,$$

for all $k$ the updates have *little influence* on frames far in the past:

$$\|\hat{\alpha}_{K-\ell|K+1} - \hat{\alpha}_{K-\ell|K}\|_2 \leq \left(\frac{\delta_*}{1 - \delta_*}\right)^\ell \|\hat{\alpha}_{K|K+1} - \hat{\alpha}_{K|K}\|_2$$
Convergence of the corrections

Under the same conditions on $D_k, E_k$,

$$1 - \delta \leq \lambda_{\text{min}}(D_k) \leq \lambda_{\text{min}}(D_k) \leq 1 + \delta, \quad \|E_k\| \leq \delta, \quad \text{for all } k$$

... meaning that

$$\|\hat{\alpha}_k|_{k+\ell} - \hat{\alpha}_{k*}\|_2 \leq C \left(\frac{\delta_*}{1 - \delta_*}\right)^\ell M_y,$$

where $\alpha_{k*}$ is the true least-squares solution and $M_y = \sup_k \|y_k\|_2$
Random samples

\[ D_k = A_k^T A_k + B_{k+1}^T B_{k+1}, \]
\[ E_{k-1} = B_k^T A_k \]

\( N = \) number of basis functions per frame bundle
\( M = \) number of samples per batch

For samples selected uniformly at random, we have with probability \( 1 - \epsilon \)

\[ 1 - \delta \leq \lambda_{\min}(D_k) \leq \lambda_{\min}(D_k) \leq 1 + \delta, \quad \|E_k\| \leq \delta, \quad \text{for fixed} \ k \]

with

\[ \delta \leq C \sqrt{\frac{N}{M} \log(N/\epsilon)} \]

so we can take

\[ M \gtrsim N \log(N/\epsilon). \]
Streaming deconvolution

shifts here go into $A_k$

shifts here go into $B_k$
Streaming $\ell_1$ Reconstruction
Optimality conditions

\[
\min_x \|x\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2
\]

- Conditions for \(x^*\) (supported on \(\Gamma^*\)) to be a solution:

\[
\phi_\gamma^T(\Phi x^* - y) = -z[\gamma] \quad \gamma \in \Gamma^*
\]

\[
|\phi_\gamma^T(\Phi x^* - y)| \leq 1 \quad \gamma \in \Gamma^{*c}
\]

where \(z[\gamma] = \text{sign}(x[\gamma])\)

- Derived simply by computing the subgradient of the functional above
Example: time-varying sparse signal

- Initial measurements. Observe
  \[ y_1 = \Phi x_1 + e_1 \]

- Initial reconstruction. Solve
  \[
  \min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y_1\|^2_2
  \]

A new set of measurements arrives:
\[ y_2 = \Phi x_2 + e_2 \]
Reconstruct again using \(\ell_1\)-min:
\[
\min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y_2\|^2_2
\]
We can gradually move from the first solution to the second solution using homotopy:
\[
\min \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - (1 - \epsilon) y_1 - \epsilon y_2\|^2_2
\]
Take \(\epsilon\) from 0 \(\to\) 1
Example: time-varying sparse signal

- Initial measurements. Observe
  \[ y_1 = \Phi x_1 + e_1 \]

- Initial reconstruction. Solve
  \[ \min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y_1\|_2^2 \]

- A new set of measurements arrives:
  \[ y_2 = \Phi x_2 + e_2 \]

- Reconstruct again using \(\ell_1\)-min:
  \[ \min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y_2\|_2^2 \]

We can gradually move from the first solution to the second solution using homotopy:

\[ \min \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - (1 - \epsilon)y_1 - \epsilon y_2\|_2^2 \]

Take \(\epsilon\) from 0 → 1
Example: time-varying sparse signal

- Initial measurements. Observe
  \[ y_1 = \Phi x_1 + e_1 \]

- Initial reconstruction. Solve
  \[
  \min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y_1\|_2^2
  \]

- A new set of measurements arrives:
  \[ y_2 = \Phi x_2 + e_2 \]

- Reconstruct again using \( \ell_1 \)-min:
  \[
  \min_x \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - y_2\|_2^2
  \]

- We can gradually move from the first solution to the second solution using homotopy
  \[
  \min \lambda \|x\|_1 + \frac{1}{2} \|\Phi x - (1 - \epsilon)y_1 - \epsilon y_2\|_2^2
  \]

  Take \( \epsilon \) from 0 \( \rightarrow \) 1
Example: time-varying sparse signal

\[
\min \lambda \|x\|_1 + \frac{1}{2} \| \Phi x - (1 - \epsilon) y_{\text{old}} - \epsilon y_{\text{new}} \|_2^2, \quad \text{take } \epsilon \text{ from } 0 \to 1
\]

- Path from old solution to new solution is \textit{piecewise linear}
- Optimality conditions for fixed \(\epsilon\):

\[
\Phi^T_{\Gamma}(\Phi x - (1 - \epsilon) y_{\text{old}} - \epsilon y_{\text{new}}) = -\lambda \sign x_{\Gamma}
\]

\[
\|\Phi^T_{\Gamma_c}(\Phi x - (1 - \epsilon) y_{\text{old}} - \epsilon y_{\text{new}})\|_\infty < \lambda
\]

\(\Gamma = \text{active support}\)
- Update direction:

\[
\partial x = \begin{cases} 
- (\Phi^T_{\Gamma} \Phi_{\Gamma})^{-1}(y_{\text{old}} - y_{\text{new}}) & \text{on } \Gamma \\
0 & \text{off } \Gamma
\end{cases}
\]
Path from old solution to new solution

\[ \Gamma = \text{support of current solution.} \]

Move in this direction

\[
\partial x = \begin{cases} 
-(\Phi^T \Phi_{\Gamma})^{-1}(y_{\text{old}} - y_{\text{new}}) & \text{on } \Gamma \\
0 & \text{off } \Gamma 
\end{cases}
\]

until support changes, or one of these constraints is violated:

\[
\left| \phi_{\gamma}^T(\Phi(x + \epsilon \partial x) - (1 - \epsilon)y_{\text{old}} - \epsilon y_{\text{new}}) \right| < \lambda \quad \text{for all } \gamma \in \Gamma^c
\]
### Numerical experiments: time-varying sparse signals

<table>
<thead>
<tr>
<th>Signal type</th>
<th>DynamicX* (nProdAtA, CPU)</th>
<th>LASSO homotopy (nProdAtA, CPU)</th>
<th>GPSR-BB (nProdAtA, CPU)</th>
<th>FPC_AS (nProdAtA, CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N = 1024, M = 512, T = m/5, k ~ T/20, Values = +/- 1</td>
<td>(23.72, 0.132)</td>
<td>(235, 0.924)</td>
<td>(104.5, 0.18)</td>
<td>(148.65, 0.177)</td>
</tr>
<tr>
<td>Blocks</td>
<td>(2.7, 0.028)</td>
<td>(76.8, 0.490)</td>
<td>(17, 0.133)</td>
<td>(53.5, 0.196)</td>
</tr>
<tr>
<td>Pcw. Poly.</td>
<td>(13.83, 0.151)</td>
<td>(150.2, 1.096)</td>
<td>(26.05, 0.212)</td>
<td>(66.89, 0.25)</td>
</tr>
<tr>
<td>House slices</td>
<td>(26.2, 0.011)</td>
<td>(53.4, 0.019)</td>
<td>(92.24, 0.012)</td>
<td>(90.9, 0.036)</td>
</tr>
</tbody>
</table>

\[ \tau = 0.01 \|A^T y\|_\infty \]

nProdAtA: roughly the avg. no. of matrix vector products with \( A \) and \( A^T \)

CPU: average cpu time to solve

[Asif and R. 2009]
Other updates

\[
\min \limits_x \| Wx \|_1 + \frac{1}{2} \| \Phi x - y \|_2^2
\]

\( W = \) weights (diagonal, positive)

Using similar ideas, we can *dynamically update* the solution when

- the underlying signal changes slightly,
- we add/remove measurements,
- the weights changes,

But none of these are really "predict and update" ...
A general, flexible homotopy framework

We want to solve

$$\min_x \|Wx\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2$$

- Initial guess/prediction: $v$
- Solve

$$\min_x \|Wx\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2 + (1 - \epsilon)u^T x$$

for $\epsilon : 0 \to 1$.
- Taking

$$u = -Wz - \Phi^T (\Phi v - y)$$

for some $z \in \partial(\|v\|_1)$ makes $v$ optimal for $\epsilon = 0$. 
Moving from the warm-start to the solution

\[
\min_x \| Wx \|_1 + \frac{1}{2} \| \Phi x - y \|_2^2 + (1 - \epsilon) u^T x
\]

The optimality conditions are

\[
\Phi^T_{\Gamma} (\Phi x - y) + (1 - \epsilon) u = -W \text{ sign } x_{\Gamma}
\]

\[
| \phi^T_{\gamma} (\Phi x - y) + (1 - \epsilon) u | \leq W[\gamma, \gamma]
\]

We move in direction

\[
\partial x = \begin{cases} 
    u_{\Gamma} & \text{on } \Gamma \\
    0 & \text{on } \Gamma^c 
\end{cases}
\]

until a component shrinks to zero or a constraint is violated, yielding new \( \Gamma \)
Streaming sparse recovery

Observations: \( y_t = \Phi_t x_t + e_t \)

Representation: \( x[n] = \sum_{p,k} \alpha_{p,k} \psi_{p,k}[n] \)
Streaming sparse recovery

Iteratively reconstruct the signal over a sliding (active) interval, form \( u \) from your prediction, then take \( \epsilon : 0 \to 1 \) in

\[
\min_{\alpha} \| W\alpha \|_1 + \frac{1}{2} \| \tilde{\Phi} \tilde{\Psi} \alpha - \tilde{y} \|_2^2 + (1 - \epsilon) u^T \alpha
\]

where \( \tilde{\Psi}, \tilde{y} \) account for edge effects.
Streaming signal recovery: Simulation

(Top-left) Mishmash signal (zoomed in for first 2560 samples. 
(Top-right) Error in the reconstruction at $R=N/M = 4$. 
(Bottom-left) LOT coefficients. (Bottom-right) Error in LOT coefficients
Streaming signal recovery: Simulation

(left) SER at different R from \( \pm 1 \) random measurements in 35 db noise
(middle) Count for matrix-vector multiplications
(right) Matlab execution time
Streaming signal recovery: Dynamic signal

Observation/evolution model:

\[ y_t = \Phi_t x_t + e_t \]
\[ x_{t+1} = F_t x_t + d_t \]

We solve

\[
\min_{\alpha} \sum_t \| W_t \alpha_t \|_1 + \frac{1}{2} \| \Phi_t \Psi_t \alpha_t - y_t \|_2^2 + \frac{1}{2} \| F_{t-1} \Psi_{t-1} \alpha_{t-1} - \Psi_t \alpha_t \|_2^2
\]

(formulation similar to Vaswani 08, Carmi et al 09, Angelosante et al 09, Zainel at al 10, Charles et al 11)

using

\[
\min_{\alpha} \| W \alpha \|_1 + \frac{1}{2} \| \Phi \tilde{\Psi} \alpha - \tilde{y} \|_2^2 + \frac{1}{2} \| \bar{F} \tilde{\Psi} \alpha - \tilde{q} \|_2^2 + (1 - \epsilon) u^T \alpha
\]
(Top-left) Piece-Regular signal (shifted copies) in image
(Top-right) Error in the reconstruction at R=N/M = 4.
(Bottom-left) Reconstructed signal at R=4.
(Bottom-right) Comparison of SER for the L1-regularized and the L2-regularized problems
Dynamic signal: Simulation

(left) SER at different R from ±1 random measurements in 35 db noise
(middle) Count for matrix-vector multiplications
(right) Matlab execution time

(Left) SER at different R from ±1 random measurements in 35 db noise
(Middle) Count for matrix-vector multiplications
(Right) Matlab execution time