Recursive Compressed Sensing

Nikolaos M. Freris, *Member, IEEE,*, Orhan Öçal, *Student member, IEEE,* and Martin Vetterli, *Fellow, IEEE*

***Abstract*—We introduce a recursive algorithm for performing compressed sensing on streaming data. The approach consists of a) *recursive encoding*, where we sample the input stream via overlapping windowing and make use of the previous measurement in obtaining the next one, and b) *recursive decoding*, where the signal estimate from the previous window is utilized in order to achieve faster convergence in an iterative optimization scheme applied to decode the new one. To remove estimation bias, a two-step estimation procedure is proposed comprising support set detection and signal amplitude estimation. Estimation accuracy is enhanced by a non-linear voting method and averaging estimates over multiple windows. We analyze the computational complexity and estimation error, and show that the normalized error variance asymptotically goes to zero for sublinear sparsity. Our simulation results show speed up of an order of magnitude over traditional CS, while obtaining significantly lower reconstruction error under mild conditions on the signal magnitudes and the noise level.**

***Index Terms*—Compressed sensing, recursive algorithms, streaming data, LASSO, machine learning, optimization, MSE.**

I. Introduction

In signal processing, it is often the case that signals of interest can be represented sparsely by using few coefficients in an appropriately selected orthonormal basis or frame. For example, the Fourier basis is used for bandlimited signals, while wavelet bases are used for piecewise continuous signals–with applications in communications for the former, and image compression for the latter. While a small number of coefficients in the respective basis may be enough for high accuracy representation, the celebrated Nyquist/Shannon sampling theorem suggests a sampling rate that is at least twice the signal bandwidth, which, in many cases, is much higher than the sufficient number of coefficients [2], [3].

The Compressed Sensing (CS)–also referred to as Compressive Sampling–framework was introduced for sampling signals not according to bandwidth, but rather to their *information content*, i.e.,, the number of degrees of freedom. This sampling paradigm suggests a lower sampling rate compared to the classical sampling theory for signals that have sparse representation in some fixed basis [2], or even non-bandlimited signals [3], to which traditional sampling does not even apply.

The foundations of CS have been developed in [4], [5]. Although the field has been extensively studied for nearly a decade, performing CS on streaming data still remains fairly open, and an efficient recursive algorithm is, to the best of our knowledge, not available. This is the topic of the current paper, where we study the architecture for compressively sampling an input data stream, and analyze the computational complexity and stability of signal estimation from noisy samples. The main contributions are:

1) We process the data stream by successively performing CS in sliding overlapping windows. Sampling overhead is minimized by recursive computations using a cyclic rotation of the same sampling matrix. A similar approach is applicable when the data are sparsely representable in the Fourier domain.
2) We perform recursive decoding of the obtained samples by using the estimate from a previously decoded window to obtain a warm-start in decoding the next window (via an iterative optimization method).
3) In our approach, a given entry of the data stream is sampled over multiple windows. In order to enhance estimation accuracy, we propose a three-step procedure to combine estimates corresponding to a given sample obtained from different windows:
   - **Support detection** amounts to estimating whether or not a given entry is non-zero. This is accomplished by a voting strategy over multiple overlapping windows containing the entry.
   - **Ordinary least-squares** is performed in each window on the determined support.
   - **Averaging** of estimates across multiple windows yields the final estimate for each entry of the data stream.
4) Extensive experiments showcase the merits of our approach in terms of substantial decrease in both run-time and estimation error.

Similar in spirit to our approach are the works of Garrigues and El Ghaoui [6], Boufounos and Asif [7] and Asif and Romberg [8]. In [6], a recursive algorithm was proposed for solving LASSO based on warm-start. In [7], the data stream is assumed sparse in the frequency domain, and *Streaming Greedy Pursuit* is proposed for progressively filtering measurements in order to reconstruct the data stream. In [8], the authors analyze the use of warm-start for speeding up the decoding step. Our work is different in that: a) it both minimizes sampling overhead by recursively encoding the data stream, as well as b) produces high-accuracy estimates by combining information over multiple windows at the decoding step.

The organization of the paper is as follows: Section II describes the notation, definitions and the related literature on CS. Section III introduces the problem formulation and describes the key components of Recursive CS (RCS): *recursive sampling* and *recursive estimation*. We analyze the proposed

The authors are with the School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland. {nikolaos.freris, orhan.ocal, martin.vetterli}@epfl.ch.

A preliminary version of this work was presented at the 51^{st} Allerton conference, 2013 [1].

method and discuss extensions in Section IV. Experimental results on resilience to noise and execution time of RCS are reported in Section VI.

II. Background

This section introduces the notation and definitions used in the paper, and summarizes the necessary background on CS.

A. Notation

Throughout the paper, we use capital boldface letters to denote matrices (e.g., $\boldsymbol{A}$) and boldface lowercase letters to denote vectors (e.g., $\boldsymbol{x}$). We use x_i to denote the i^{th} entry of vector $\boldsymbol{x}$, $\boldsymbol{a}_i$ to denote the i^{th} column of matrix $\boldsymbol{A}$, and A_{ij} to denote its (i,j) entry . The i^{th} sampling instance (e.g., i^{th} window of the input stream, i^{th} sampling matrix, i^{th} sample) is denoted by superscript (e.g., $\boldsymbol{x}^{(i)}$, $\boldsymbol{A}^{(i)}$, $\boldsymbol{y}^{(i)}$). The cardinality of a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$, and we use $\{x_i\}$ as shorthand notation for the infinite sequence $\{x_i\}_{i=0,1,\ldots}$. Last, we use $\mathbb{E}_x[\,\cdot\,]$ to denote the conditional expectation $\mathbb{E}_x[\,\cdot\,] = \mathbb{E}[\,\cdot\,|x]$.

B. Definitions and Properties

In the following, we summarize the key definitions related to compressed sensing.

Definition 1 (κ-sparsity)**.** *For a vector $\boldsymbol{x} \in \mathbb{R}^n$ we define the support* $\mathrm{supp}(\boldsymbol{x}) := \{i : x_i \neq 0\}$. *The ℓ_0 pseudonorm is $\|\boldsymbol{x}\|_0 := |supp(\boldsymbol{x})|$. We say that a vector $\boldsymbol{x}$ is* κ-sparse *if and only if $\|\boldsymbol{x}\|_0 \leq \kappa$.*

Definition 2 (Mutual Coherence)**.** *For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, the mutual coherence is defined as the largest normalized inner product between any two different columns of $\boldsymbol{A}$ [9]:*

$$\mu(\boldsymbol{A}) := \max_{\substack{0\leq i,j\leq n-1 \\ i\neq j}} \frac{|\boldsymbol{a}_i^\top \boldsymbol{a}_j|}{\|\boldsymbol{a}_i\|_2 \cdot \|\boldsymbol{a}_j\|_2}. \tag{1}$$

(2)

Definition 3 (Restricted Isometry Property)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. For given $0 < \kappa < n$, the matrix $\boldsymbol{A}$ is said to satisfy the Restricted Isometry Property (RIP) if there exists $\delta_\kappa \in [0,1]$ such that:*

$$(1-\delta_\kappa)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta_\kappa)\|\boldsymbol{x}\|_2^2 \tag{3}$$

holds for all $\boldsymbol{x} \in \mathbb{R}^n$ κ-sparse vectors, for a constant $\delta_\kappa \geq 0$ sufficiently small [2].

The value δ_κ is called the *restricted isometry constant* of $\boldsymbol{A}$ for κ-sparse vectors. Evidently, an equivalent description of RIP is that every subset of κ columns of $\boldsymbol{A}$ approximately behaves like an orthonormal system [10], hence $\boldsymbol{A}\boldsymbol{x}$ is approximately an isometry for κ-sparse vectors.

Unfortunately, RIP is NP-hard even to verify for a given matrix as it requires $\binom{n}{\kappa}$ eigendecompositions. The success story lies in that properly constructed random matrices satisfy RIP with overwhelming probability [2], for example:

1) Sampling n random vectors uniformly at random from the m-dimensional unit sphere [2].
2) Random partial Fourier matrices obtained by selecting m rows from the n dimensional Fourier matrix F uniformly at random, where:

$$\boldsymbol{F} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \omega & \omega^2 & \ldots & \omega^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^n & \omega^{2n} & \ldots & \omega^{(n-1)^2} \end{bmatrix}$$

   for $\omega = e^{i2\pi/N}$.
3) Random Gaussian matrices with entries drawn i.i.d. from $\mathcal{N}(0, 1/m)$.
4) Random Bernoulli matrices with

$$A_{i,j} \in \left\{1/\sqrt{m}, -1/\sqrt{m}\right\}$$

   with equal probability, or [11]:

$$A_{ij} = \begin{cases} 1 & \text{with probability } \frac{1}{6}, \\ 0 & \text{with probability } \frac{2}{3}, \\ -1 & \text{with probability } \frac{1}{6}. \end{cases} \tag{4}$$

   having the added benefit of providing a *sparse* sampling matrix.

For the last two cases, $\boldsymbol{A}$ satisfies a prescribed δ_κ for any $\kappa \leq c_1 m/\log(n/\kappa)$ with probability $p \geq 1 - 2e^{-c_2 m}$, where constants c_1 and c_2 depend only on δ_κ [12]. The important result here is that such matrix constructions are *universal*–in the sense that they satisfy RIP which is a property that does not depend on the underlying application– as well as efficient, as they only require random number generation.

It is typically the case that $\boldsymbol{x}$ is not itself sparse, but is sparsely representable in a given orthonormal basis. In such case, we write $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{\alpha}$, where now $\boldsymbol{\alpha}$ is sparse. Compressed sensing then amounts to designing a *sensing matrix* $\bar{\boldsymbol{\Phi}} \in \mathbb{R}^{m\times n}$ such that $\boldsymbol{A} := \bar{\boldsymbol{\Phi}}\boldsymbol{\Psi}$ is a CS matrix. Luckily, random matrix constructions can still serve this purpose.

Definition 4 (**Coherence**)**.** *Let $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$ be two orthonormal bases in $\mathbb{R}^n$. The* coherence *between these two bases is defined as [2]:*

$$\mathcal{M}(\boldsymbol{\Phi}, \boldsymbol{\Psi}) := \sqrt{n} \max_{1\leq k,j\leq n} |\langle \phi_k, \psi_j \rangle|. \tag{5}$$

It follows from elementary linear algebra that $1 \leq \mathcal{M}(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \leq \sqrt{n}$ for any choice of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$.

The basis $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$ is called the *sensing* basis, while $\boldsymbol{\Psi} \in \mathbb{R}^{n\times n}$ is the *representation* basis. Compressed sensing results apply for low coherence pairs [5]. A typical example of such pairs is the Fourier and canonical basis, for which the coherence is 1 (*maximal incoherence*). Most notably, a random basis $\boldsymbol{\Phi}$ (generated by any of the previously described distributions for $m = n$), when orthonormalized is incoherent with any given basis $\boldsymbol{\Psi}$ (with high probability, $\mathcal{M}(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \approx \sqrt{2\log n}$) [2]. The sensing matrix $\bar{\boldsymbol{\Phi}}$ can be selected as a row-subset of $\boldsymbol{\Phi}$, therefore, designing a sensing matrix is not different than for the case where $\boldsymbol{x}$ is itself sparse, i.e., $\boldsymbol{\Psi} = \boldsymbol{I}_{n\times n}$, the identity matrix. For ease of presentation, we assume in the sequel that $\boldsymbol{\Psi} = \boldsymbol{I}_{n\times n}$, unless otherwise specified, but the results also hold for the general case.

C. Setting

Given linear measurements of vector $\boldsymbol{x} \in \mathbb{R}^n$

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}, \tag{6}$$

$\boldsymbol{y} \in \mathbb{R}^m$ is the vector of obtained samples and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is the *sampling (sensing)* matrix. Our goal is to recover $\boldsymbol{x}$ when $m << n$. This is an underdetermined linear system, so for a general vector $\boldsymbol{x}$ it is essentially *ill-posed*. The main result in CS is that if $\boldsymbol{x}$ is κ-sparse and $\kappa < Cm/\log(n/k)$, this is *possible*. Equivalently, random linear measurements can be used for compressively **encoding** a sparse signal, so that it is then possible to reconstruct it in a subsequent decoding step. The key concept here is that encoding is *universal*: while it is straightforward to compress if one knows the positions of the non-zero entries, the CS approach works for *all* sparse vectors, without requiring prior knowledge of the non-zero positions.

To this end, searching for the sparsest vector $\boldsymbol{x}$ that leads to the measurement $\boldsymbol{y}$, one needs to solve

$$\begin{aligned} \min \quad & \|\boldsymbol{x}\|_0 \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \end{aligned} \tag{P_0}$$

Unfortunately this problem is, in general, NP-hard requiring search over all subsets of columns of A [12], e.g., checking $\binom{n}{\kappa}$ linear systems for a solution in the worst case.

D. Algorithms for Sparse Recovery

Since (P_0) may be computationally intractable for large instances, one can seek to 'approximate' it by other tractable methods. In this section, we summarize several algorithms used for recovering sparse vectors from linear measurements, at the **decoding** phase, with provable performance guarantees.

1) Basis Pursuit: Candès and Tao [10] have shown that solving (P_0) is equivalent to solving the ℓ_1 minimization problem

$$\begin{aligned} \min \quad & \|\boldsymbol{x}\|_1 \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \end{aligned} \tag{BP}$$

for all κ-sparse vectors $\boldsymbol{x}$, if $\boldsymbol{A}$ satisfies RIP with $\delta_{2\kappa} < \sqrt{2}-1$. The optimization problem (BP) is called *Basis Pursuit*. Since the problem can be recast as a linear program, solving (BP) is computationally efficient, e.g., via interior-point methods [13], even for large problem instances as opposed to solving (P_0) whose computational complexity may be prohibitive.

2) Orthogonal Matching Pursuit: Orthogonal Matching Pursuit (OMP) is a greedy algorithm that seeks to recover sparse vectors $\boldsymbol{x}$ from noiseless measurement $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. The algorithm outputs a subset of columns of A, via iteratively selecting the column minimizing the residual error of approximating $\boldsymbol{y}$ by projecting to the linear span of previously selected columns.

Assuming $\boldsymbol{x}$ is κ-sparse, the resulting measurement $\boldsymbol{y}$ can be represented as the sum of at most κ columns of $\boldsymbol{A}$ weighted by the corresponding nonzero entries of $\boldsymbol{x}$. Let the columns of $\boldsymbol{A}$ be normalized to have unit ℓ_2-norm. For iteration index t, let $\boldsymbol{r}_t$ denote the residual vector, let $\boldsymbol{c}_t \in \mathbb{R}^t$ be the solution to the least squares problem at iteration t, S_t set of indices and $\boldsymbol{A}_{S_t}$ the submatrix obtained by extracting columns of $\boldsymbol{A}$ indexed by S_t. The OMP algorithm operates as follows: Initially set $t = 1$, $\boldsymbol{r}_0 = \boldsymbol{y}$ and $S_0 = \varnothing$. At each iteration the index of the column of $\boldsymbol{A}$ having highest inner product with the residual vector, i.e., $s_t = \arg\max_i \langle \boldsymbol{r}_{t-1}, \boldsymbol{a}_i \rangle$ is added to the index set, yielding $S_t = S_{t-1} \cup \{s_t\}$. Since one index is added to S_t at each iteration the cardinality is $|S_t| = t$. Then, the least squares problem

$$\boldsymbol{c}_t = \arg\min_{\boldsymbol{c} \in \mathbb{R}^t} \|\boldsymbol{y} - \sum_{j=1}^{t} c_j a_{s_j}\|_2$$

is solved in each iteration; a closed form solution is:

$$\boldsymbol{c}_t = \left(\boldsymbol{A}_{S_t}^\top \boldsymbol{A}_{S_t}\right)^{-1} \boldsymbol{A}_{S_t}^\top \boldsymbol{y},$$

and the residual vector is updated by $\boldsymbol{r}_t = \boldsymbol{y} - \sum_{j=1}^{t} c_{tj} a_{s_j}$. With $\boldsymbol{r}_t$ obtained as such, the residual vector at the end of iteration t is made orthogonal to all the vectors in the set $\{\boldsymbol{a}_i : i \in S_t\}$.

The algorithm stops when a desired stopping criterion is met, such as $\|\boldsymbol{y} - \boldsymbol{A}_{S_t}\boldsymbol{c}_t\|_2 \leq \gamma$ for some threshold $\gamma \geq 0$. Despite its simplicity, there are guarantees for *exact recovery*; OMP recovers *any* κ-sparse signal exactly if the mutual coherence of the measurement matrix $\boldsymbol{A}$ satisfies $\mu(\boldsymbol{A}) < \frac{1}{2\kappa - 1}$ [14]. Both BP and OMP handle the case of *noiseless* measurements. However, in most practical scenaria, noisy measurements are inevitable, and we address this next.

3) Least Absolute Selection and Shrinkage Operator (LASSO): Given measurements of vector $\boldsymbol{x} \in \mathbb{R}^n$ corrupted by additive noise:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}, \tag{7}$$

one can solve a relaxed version of BP, where the equality constraint is replaced by inequality to account for measurement noise:

$$\begin{aligned} \min \quad & \|\boldsymbol{x}\|_1 \\ \text{s.t.} \quad & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \tilde{\sigma}, \end{aligned} \tag{8}$$

This is best known as Least Absolute Selection and Shrinkage Operator (LASSO) in the statistics literature [15]. The value of $\tilde{\sigma}$ is selected to satisfy $\tilde{\sigma} \geq \|\boldsymbol{w}\|_2$.

By duality, the problem can be posed equivalently [13] as an unconstrained ℓ_1-regularized least squares problem:

$$\min \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{x}\|_1, \tag{9}$$

where λ is the regularization parameter that controls the trade-off between sparsity and reconstruction error. Still by duality, an equivalent version is given by the following constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \\ \text{subject to} \quad & \|\boldsymbol{x}\|_1 \leq \mu. \end{aligned} \tag{10}$$

Remark 1 (Equivalent forms of LASSO)**.** *All these problems can be made equivalent–in the sense of having the same solution set–for particular selection of parameters $(\tilde{\sigma}, \lambda, \mu)$. This can be seen by casting between the optimality conditions for each problem; unfortunately the relations obtained depend on the optimal solution itself, so there is no analytic formula for selecting a parameter from the tuple $(\tilde{\sigma}, \lambda, \mu)$ given another*

one. In the sequel, we refer to both (8), (9) *as LASSO; the distinction will be made clear from the context.*

The following theorem characterizes recovery accuracy in the noisy case through LASSO.

Theorem II.1 (Error of LASSO [2])**.** *If $\boldsymbol{A}$ satisfies RIP with $\delta_{2\kappa} < \sqrt{2}-1$, the solution $\boldsymbol{x}_*$ to* (8) *obeys:*

$$\|\boldsymbol{x}_* - \boldsymbol{x}\|_2 \leq c_0 \cdot \|\boldsymbol{x} - \boldsymbol{x}_\kappa\|_1/\sqrt{\kappa} + c_1 \cdot \tilde{\sigma}, \qquad (11)$$

for constants c_0 and c_1, where $\boldsymbol{x}_\kappa$ is the vector $\boldsymbol{x}$ with all but the largest κ components set to 0.

Theorem II.1 states that the reconstruction error is upper bounded by the sum of two terms: the first is the error due to *model mismatch*, and the second is proportional to the *measurement noise variance*. In particular, if $\boldsymbol{x}$ is κ-sparse and $\delta_{2\kappa} < \sqrt{2}-1$ then $\|\boldsymbol{x}_* - \boldsymbol{x}\|_2 \leq c_1 \cdot \tilde{\sigma}$. Additionally, for noiseless measurements $\boldsymbol{w} = \boldsymbol{0} \implies \tilde{\sigma} = 0$, we retrieve the success of BP as a special case (note that the requirement on the restricted isometry constant is identical). This assumption is satisfied with high probability by matrices obtained from random vectors sampled from the unit sphere, random Gaussian matrices and random Bernoulli matrices if $m \geq C\kappa\log(n/\kappa)$, where C is a constant depending on each instance [2]; typical values for the constants C_0 and C_1 can be found in [2] and [5], where it is proven that $C_0 \leq 5.5$ and $C_1 \leq 6$ for $\delta_{2k} = 1/4$. A different approach was taken in [16], where the replica method was used for analyzing the mean-squared error of LASSO.

The difficulty of solving (P_0) lies in estimating the *support* of vector $\boldsymbol{x}$, i.e., the positions of the non-zero entries. One may assume that solving LASSO may give some information on support, and this is indeed the case [17]. To state the result on support detection we define the *generic* κ-sparse model.

Definition 5 (Generic κ-sparse model)**.** *Let $\boldsymbol{x} \in \mathbb{R}^n$ denote a κ-sparse signal and $I_{\boldsymbol{x}} := supp(\boldsymbol{x})$ be its support set, $supp(\boldsymbol{x}) := \{i : x_i \neq 0\}$. Signal $\boldsymbol{x}$ is said to be generated by* generic κ-sparse model *if:*

1) *Support $I_{\boldsymbol{x}} \subset \{1, 2, \ldots, n\}$ of $\boldsymbol{x}$ is selected uniformly at random, and $|I_x| = \kappa$.*
2) *Conditioned on $I_{\boldsymbol{x}}$, the signs of the non zero elements are independent and equally likely to be -1 and 1.*

Theorem II.2 (Support Detection [17])**.** *Assume $\mu(\boldsymbol{A}) \leq c_1/\log n$ for some constant $c_1 > 0$, $\boldsymbol{x}$ is generated from* generic κ-sparse model*, $\kappa \leq c_2 n/(\|\boldsymbol{A}\|_2^2 \log n)$ for some constant $c_2 > 0$ and $\boldsymbol{w} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$. If $\min_{i\in I_{\boldsymbol{x}}} |x_i| > 8\sigma\sqrt{2\log n}$, the LASSO estimate obtained by choosing $\lambda = 4\sigma\sqrt{2\log n}$ satisfies:*

$$\begin{aligned} \mathrm{supp}(\hat{\boldsymbol{x}}) &= \mathrm{supp}(\boldsymbol{x}) \\ \mathrm{sgn}(\hat{x}_i) &= \mathrm{sgn}(x_i), \quad \forall i \in I_{\boldsymbol{x}} \end{aligned}$$

with probability at least $1 - \frac{2}{n}\left(\frac{1}{\sqrt{2\pi\log n}} + \frac{|I_{\boldsymbol{x}}|}{n}\right) - O\left(\frac{1}{n^{2\log 2}}\right)$ and

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2^2 \leq c_3\kappa(\log n)\sigma^2,$$

with probability at least $1 - 6n^{-2\log 2} - n^{-1}(2\pi\log n)^{-1/2}$, for some positive constant c_3.

Another result on near support detection, or as alternatively called *ideal model selection* for LASSO is given in [18] based on the so called *irrepresentable* condition of the sampling matrix introduced therein.

Remark 2 (Algorithms for LASSO)**.** *There is a wealth of numerical methods for LASSO stemming from convex optimization. LASSO is a convex program (in all equivalent forms) and the unconstrained problem* (9) *can be easily recast as a quadratic program, which can be handled by interior point methods [19]. This is the case when using a generalized convex solver such as cvx [20]. Additionally, iterative algorithms have been developed specifically for LASSO; all these are inspired by proximal methods [21] for non-smooth convex optimization: FISTA [22] and SpaRSA [23] are accelerated proximal gradient methods [21], SALSA [24] is an application of the alternative direction method of multipliers. These methods are* first-order *methods [19], in essence generalizations of the gradient method. For error defined as $G(\boldsymbol{x}_{[t]}) - G(\boldsymbol{x}_*)$ where $G(\boldsymbol{x})$ is the objective function of LASSO in* (9)*, $\boldsymbol{x}_{[t]}$ is the estimate at iteration number t and $\boldsymbol{x}_* = \mathrm{argmin}_{\boldsymbol{x}} G(\boldsymbol{x})$ is the optimal soultion, the error decays as $1/t^2$ for FISTA, SpaRSA and SALSA. Recently, a proximal Newton-type method was devised for LASSO [25] with substantial speedup; the convergence rate is globally no worse than $1/t^2$, but is locally* quadratic *(i.e., goes to zero roughly like e^{-ct^2}).*

Remark 3 (Computational complexity)**.** *In iterative schemes, computational complexity is considered at a per-iteration basis: a) interior-point methods require solving a* dense *linear system, hence a cost of $O(n^3)$ per iteration, b) first-order proximal methods only perform matrix-vector multiplications at a cost of $O(n^2)$, while the second-order method proposed in [25] requires solving a* sparse *linear system at a resulting cost of $O(\kappa^3)$. The total complexity depends also on the number of iterations until convergence; we analyze this in Sec. V-B. Note that the cost of decoding dominates that of encoding which requires a single matrix-vector multiplication, i.e., $O(mn)$ operations.*

Our approach is generic, in that it does not rely on a particular selection of numerical solver. It uses *warm-start* for accelerated convergence, so using an algorithm like [25] may yield improvements over the popular FISTA that we currently use in experiments.

We conclude this section by providing optimality conditions for LASSO, which can serve in determining termination criteria for iterative optimization algorithms. We show the case of unconstrained LASSO, but similar conditions hold for the constrained versions (8), (10).

Remark 4 (Optimality conditions for LASSO)**.** *For unconstrained LASSO cf.* (9)*, define $\boldsymbol{x}^*$ to be an optimal solution, and $\boldsymbol{g} := \boldsymbol{A}^\top(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^*)$. The necessary and sufficient KKT*

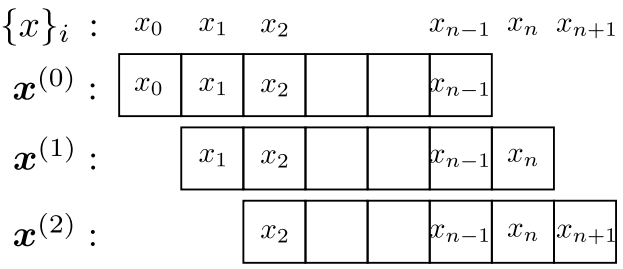


Fig. 1: Illustration of the overlapping window processing for the data stream $\{\boldsymbol{x}^{(i)}\}_{i=0,1\ldots}$.

conditions for optimality [19] are:

$$\begin{aligned} g_i &= \frac{\lambda}{2}\,\mathrm{sgn}(x_i^*) \quad \textit{for}\,\{i : x_i^* \neq 0\} \\ |g_j| &< \frac{\lambda}{2} \qquad\qquad \textit{for}\,\{j : x_j^* = 0\}\,. \end{aligned} \tag{12}$$

As termination criterion, we use ϵ-optimality, for some $\epsilon > 0$ suffieciently small:

$$\begin{aligned} |g_i - \frac{\lambda}{2}\,\mathrm{sgn}(x_i^*)| &\leq \epsilon \qquad \text{for}\,\{i : x_i^* \neq 0\} \\ |g_j| &< \frac{\lambda}{2} + \epsilon \quad \text{for}\,\{j : x_j^* = 0\}\,. \end{aligned} \tag{13}$$

III. Recursive Compressed Sensing

We consider the case that the signal of interest is an infinite sequence, $\{x_i\}_{i=0,1,\ldots}$, and process the input stream via successive windowing; we define

$$\boldsymbol{x}^{(i)} := \begin{bmatrix} x_i & x_{i+1} & \ldots & x_{i+n-1} \end{bmatrix}^\top \tag{14}$$

to be the i^{th} window taken from the streaming signal. If $\boldsymbol{x}^{(i)}$ is known to be sparse, one can apply the tools surveyed in Section II to recover the signal portion in each window, hence the data stream. However, the involved operations are costly and confine an efficient online implementation.

In this section, we present our approach to compressively sampling streaming data, based on recursive **encoding-decoding**. The proposed method has low complexity in both the sampling and estimation parts which makes the algorithm suitable for an online implementation.

A. Problem Formulation

From the definition of $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$ we have:

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \boldsymbol{x}^{(i-1)} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} x_{i+n-1}, \tag{15}$$

which is in the form of a $n-$dynamical system with scalar input. The sliding window approach is illustrated in Fig. 1.

Our goal is to design a robust low-complexity sliding-window algorithm which provides estimates $\{\hat{x}_i\}$ using successive measurements $\boldsymbol{y}^{(i)}$ of the form

$$\boldsymbol{y}^{(i)} = \boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{w}^{(i)}, \tag{16}$$

where $\{\boldsymbol{A}^{(i)}\}$ is a sequence of measurement matrices. This is possible if $\{x_i\}$ is sufficiently sparse in each window, namely if $\|\boldsymbol{x}^{(i)}\|_0 \leq \kappa$ for each i, where $\kappa << n$ (or if this holds with sufficiently high probability), and $\{\boldsymbol{A}^{(i)}\}$ are CS matrices, i.e., satisfy the RIP as explained in the prequel.

Note that running such an algorithm online is costly and therefore, it is integral to design an alternative to an ad-hoc method. We propose an approach that leverages the signal overlap between successive windows, consisting of recursive sampling and recursive estimation.

Recursive Sampling: To avoid a full matrix-vector multiplication for each $\boldsymbol{y}^{(i)}$, we design $\boldsymbol{A}^{(i)}$ so that we can reuse $\boldsymbol{y}^{(i)}$ in computing $\boldsymbol{y}^{(i+1)}$ with low computation overhead, or

$$\boldsymbol{y}^{(i+1)} = f\left(\boldsymbol{y}^{(i)}, x_{i+n}, x_i\right).$$

Recursive Estimation: In order to speed up the convergence of an iterative optimization scheme, we make use of the estimate corresponding to the previous window, $\hat{\boldsymbol{x}}^{(i-1)}$, to derive a starting point, $\hat{\boldsymbol{x}}^{(i)}_{[0]}$, for estimating $\hat{\boldsymbol{x}}^{(i)}$, or

$$\hat{\boldsymbol{x}}^{(i)}_{[0]} = g\left(\hat{\boldsymbol{x}}^{(i-1)}, \boldsymbol{y}^{(i)}\right).$$

B. Recursive Sampling of sparse signals

We propose the following recursive sampling scheme with low computational overhead; it reduces the complexity to $O(m)$ vs. $O(mn)$ as required by the standard data encoding.

We derive our scheme for the most general case of noisy measurements, with ideal measurements following as special case.

At the first iteration, there is no prior estimate, so we necessarily have to compute

$$\boldsymbol{y}^{(0)} = \boldsymbol{A}^{(0)}\boldsymbol{x}^{(0)} + \boldsymbol{w}^{(0)}.$$

We choose a sequence of sensing matrices $\boldsymbol{A}^{(i)}$ recursively as:

$$\boldsymbol{A}^{(i+1)} = \begin{bmatrix} \boldsymbol{a}_1^{(i)} & \boldsymbol{a}_2^{(i)} & \ldots & \boldsymbol{a}_{n-1}^{(i)} & \boldsymbol{a}_0^{(i)} \end{bmatrix} = \boldsymbol{A}^{(i)}\boldsymbol{P} \tag{17}$$

where $\boldsymbol{a}_j^{(i)}$ is the j^{th} column of $\boldsymbol{A}^{(i)}$–where we have used the convention $j \in \{0, 1, \ldots, n-1\}$ for notational convenience– and $\boldsymbol{P}$ is a permutation matrix:

$$\boldsymbol{P} := \begin{bmatrix} 0 & \ldots & 0 & 1 \\ 1 & \ldots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & 1 & 0 \end{bmatrix}. \tag{18}$$

The success of this data encoding scheme is ensured by noting that if $\boldsymbol{A}^{(0)}$ satisfies RIP for given κ with constant δ_κ, then $\boldsymbol{A}^{(i)}$ satisfies RIP for the same κ, δ_κ, due to the fact that RIP is insensitive to permutations of the columns of $\boldsymbol{A}^{(0)}$.

Given the particular recursive selection of $\boldsymbol{A}^{(i)}$ we can

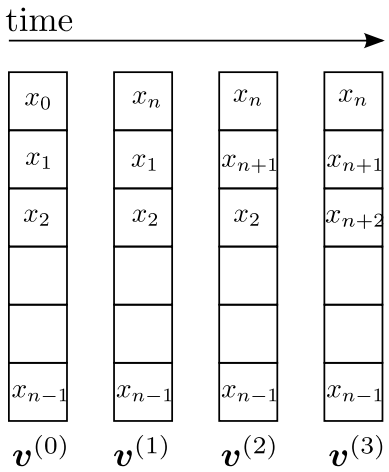


Fig. 2: Illustration of $\boldsymbol{v}^{(i)}$ for the first four windows. The sampling of $\boldsymbol{x}^{(i)}$ by the matrix $\boldsymbol{A}^{(i)}$ is equivalent to sampling $\boldsymbol{v}^{(i)}$ by $\boldsymbol{A}^{(0)}$, i.e., $\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} = \boldsymbol{A}^{(0)}\boldsymbol{v}^{(i)}$.

compute $\boldsymbol{y}^{(i+1)}$ recursively as:

$$\begin{aligned}\boldsymbol{y}^{(i+1)} &= \boldsymbol{A}^{(i+1)}\boldsymbol{x}^{(i+1)} + \boldsymbol{w}^{(i+1)} \\ &= \boldsymbol{A}^{(i)}\boldsymbol{P}\boldsymbol{x}^{(i+1)} + \boldsymbol{w}^{(i+1)} \\ &= \boldsymbol{A}^{(i)}\left(\boldsymbol{x}^{(i)} + \begin{bmatrix}1\\ \boldsymbol{0}_{n-1}\end{bmatrix}(x_{i+n} - x_i)\right) + \boldsymbol{w}^{(i+1)} \\ &= \boldsymbol{y}^{(i)} + (x_{i+n} - x_i)\boldsymbol{a}_1^{(i)} + \boldsymbol{w}^{(i+1)} - \boldsymbol{w}^{(i)}, \end{aligned} \tag{19}$$

where $\boldsymbol{0}_{n-1}$ denotes the all 0 vector of length $n-1$. This takes the form of a noisy *rank*-1 update:

$$\boldsymbol{y}^{(i+1)} = \boldsymbol{y}^{(i)} + \underbrace{(x_{i+n} - x_i)\boldsymbol{a}_1^{(i)}}_{\text{rank-1 update}} + \boldsymbol{z}^{(i+1)}, \tag{20}$$

where the *innovation* is the scalar difference between the new sampled portion of the stream, namely x_{i+n}, and the entry x_i that belongs in the previous window but not in the current one. Above, we also defined $\boldsymbol{z}^{(i)} := \boldsymbol{w}^{(i)} - \boldsymbol{w}^{(i-1)}$ to be the noise increment; note that the noise sequence $\{\boldsymbol{z}^{(i)}\}$ has independent entries if $\boldsymbol{w}^{(i)}$ is an independent increment process. Our approach naturally extends to sliding the window by $1 < \tau \leq n$ units, in which case we have a rank-τ update, cf. Sec. IV

Remark 5. *The particular selection of the sampling matrices* $\{\boldsymbol{A}^{(i)}\}_{i=0,1,\ldots}$ *given in* (17) *satisfies* $\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} = \boldsymbol{A}^{(0)}\boldsymbol{P}^i\boldsymbol{x}^{(i)}$. *Defining*

$$\boldsymbol{v}^{(i)} := \boldsymbol{P}^i\boldsymbol{x}^{(i)}, \tag{21}$$

recursive sampling can be viewed as encoding $\boldsymbol{v}^{(i)}$ *by using the same measurement matrix* $\boldsymbol{A}^{(0)}$. *With the particular structure of* $\boldsymbol{x}^{(i)}$ *given in* (14)*, all of the entries of* $\boldsymbol{v}^{(i)}$ *and* $\boldsymbol{v}^{(i-1)}$ *are equal except* $v_{i-1}^{(i)}$. *Thus the resulting problem can be viewed as signal estimation with partial information.*

1) Recursive sampling in Orthonormal Basis: So far, we have addressed the case that for a given $n \in \mathbb{Z}^+$, a given window $\boldsymbol{x}^{(i)}$ of length n obtained from the sequence $\{x_i\}$ is κ-spase: $\|\boldsymbol{x}^{(i)}\|_0 \leq \kappa$, $\forall i$. In general, it might rarely be the case that $\boldsymbol{x}^{(i)}$ is sparse itself, however it may be sparse when represented in a properly selected basis (for instance the Fourier basis for time series or a wavelet basis for images). We show the generalization below.

Let $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$ be sparsely representable in a given orthonormal basis $\boldsymbol{\Psi}$, i.e., $\boldsymbol{x}^{(i)} = \boldsymbol{\Psi}\boldsymbol{\alpha}^{(i)}$, where $\boldsymbol{\alpha}^{(i)}$ is sparse. Assuming a common basis for the entire sequence $\{x_i\}$ (over windows of size n) we have:

$$\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} = \boldsymbol{A}^{(i)}\boldsymbol{\Psi}\boldsymbol{\alpha}^{(i)}.$$

For the CS encoding/decoding procedure to carry over, we need that $\boldsymbol{A}^{(i)}\boldsymbol{\Psi}$ satisfy RIP. The key result here is that RIP is satisfied with high probability for the product of a random matrix $\boldsymbol{A}^{(i)}$ and any fixed matrix [12]. In this case the LASSO problem is expressed as:

$$\text{minimize} \quad \|\boldsymbol{A}^{(i)}\boldsymbol{\Psi}\boldsymbol{\alpha}^{(i)} - \boldsymbol{y}^{(i)}\|_2^2 + \lambda\|\boldsymbol{\alpha}^{(i)}\|_1,$$

where the input signal is expressed as $\boldsymbol{x}^{(i)} = \boldsymbol{\Psi}\boldsymbol{\alpha}^{(i)}$, and measurements are still given by $\boldsymbol{y}^{(i)} = \boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{w}^{(i)}$.

Lemma III.1 (Recursive Sampling in Orthonormal Basis)**.** *Let* $\boldsymbol{x}^{(i)} = \boldsymbol{\Psi}\alpha^{(i)}$*, where* $\boldsymbol{\Psi}$ *is an orthonormal matrix with inverse* $\boldsymbol{\Gamma} := \boldsymbol{\Psi}^{-1}$. *Then,*

$$\begin{aligned}\boldsymbol{\alpha}^{(i+1)} =& \boldsymbol{\Gamma}\boldsymbol{\Pi}\boldsymbol{\Psi}\boldsymbol{\alpha}^{(i)} \\ &+ \boldsymbol{\gamma}_{n-1}\left(\boldsymbol{\psi}_{(n-1)}\boldsymbol{\alpha}^{(i+1)} - \boldsymbol{\psi}_{(0)}\boldsymbol{\alpha}^{(i)}\right), \end{aligned} \tag{22}$$

where $\boldsymbol{\Pi} := \boldsymbol{P}^\top$*, and* $\boldsymbol{\gamma}_{(0)}$ *and* $\boldsymbol{\gamma}_{(n-1)}$ *denote the first and last row of* $\boldsymbol{\Gamma}$*, respectively.*

Proof. By the definition of $\boldsymbol{x}^{(i+1)}$ we have:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{\Pi}\boldsymbol{x}^{(i)} + \begin{bmatrix}\boldsymbol{0}_{n-1}\\ 1\end{bmatrix}(x_{i+n} - x_i).$$

Since $\boldsymbol{x}^{(i)} = \boldsymbol{\Psi}\boldsymbol{\alpha}^{(i)}$, it holds:

$$\begin{aligned} x_i = x_0^{(i)} &= \begin{bmatrix}1 & \boldsymbol{0}_{n-1}\end{bmatrix}\boldsymbol{\Psi}\boldsymbol{\alpha}^{(i+1)} \\ x_{i+n} = x_{n-1}^{(i+1)} &= \begin{bmatrix}\boldsymbol{0}_{n-1} & 1\end{bmatrix}\boldsymbol{\Psi}\boldsymbol{\alpha}^{(i+1)}. \end{aligned}$$

Using these equations along with $\boldsymbol{\alpha}^{(i)} = \boldsymbol{\Gamma}\boldsymbol{x}^{(i)}$ yields:

$$\begin{aligned}\boldsymbol{\alpha}^{(i+1)} &= \boldsymbol{\Gamma}\boldsymbol{x}^{(i+1)} = \boldsymbol{\Gamma}\boldsymbol{\Pi}\boldsymbol{x}^{(i)} + \boldsymbol{\Gamma}\begin{bmatrix}\boldsymbol{0}_{n-1}\\ 1\end{bmatrix}(x_{i+n} - x_i) \\ &= \boldsymbol{\Gamma}\boldsymbol{\Pi}\boldsymbol{x}^{(i)} + (x_{i+n} - x_i)\boldsymbol{\gamma}_{n-1} \\ &= \boldsymbol{\Gamma}\boldsymbol{\Pi}\boldsymbol{\Psi}\boldsymbol{\alpha}^{(i)} + \boldsymbol{\gamma}_{n-1}\left((\boldsymbol{\psi}_{n-1}^\top)^\top\boldsymbol{\alpha}^{(i+1)} - (\boldsymbol{\psi}_0^\top)^\top\boldsymbol{\alpha}^{(i)}\right). \end{aligned}$$

□

2) Recursive Sampling in Fourier Basis: The Fourier basis is of particular interest in many practical applications, e.g., time-series analysis. For such a basis, an efficient update rule can be derived, as is shown in the next corollary.

Corollary III.2. *Let* $\boldsymbol{\Psi}$ *be* $n \times n$ *inverse Discrete Fourier Transform (IDFT) matrix with entries* $\Psi_{i,j} = \omega^{ij}/\sqrt{n}$ *where* $i, j \in \{0, \ldots, n-1\}$ *and* $\omega := e^{j\frac{2\pi}{n}}$. *In such case:*

$$\boldsymbol{\alpha}^{(i+1)} = \boldsymbol{\Omega}_n\boldsymbol{\alpha}^{(i)} + \boldsymbol{f}_{n-1}\left(\boldsymbol{\psi}_{(n-1)}\boldsymbol{\alpha}^{(i+1)} - \boldsymbol{\psi}_{(0)}\boldsymbol{\alpha}^{(i)}\right) \tag{23}$$

where $\boldsymbol{\Omega}_n$ *is the* $n \times n$ *diagonal matrix with* $(\Omega_n)_{i,i} = \omega^{-i}$*, and* $\boldsymbol{F} = \boldsymbol{\Psi}^{-1}$ *is the orthonormal Fourier basis.*

Proof. Circular shift in the time domain corresponds to multiplication by complex exponentials in the Fourier domain, i.e., $\boldsymbol{F\Pi} = \boldsymbol{\Omega}_n \boldsymbol{F}$, and the result follows from $\boldsymbol{F\Psi} = \boldsymbol{I}$. □

Remark 6 (Complexity of recursive sampling in an orthonormal basis)**.** *In general, the number of computations for calculating $\boldsymbol{\alpha}^{(i+1)}$ from $\boldsymbol{\alpha}^{(i)}$ is $O(n^2)$. For the particular case of using Fourier basis, the complexity is reduced to only $O(n)$, i.e., we have* zero-overhead *for sampling directly on the Fourier domain.*

C. Recursive Estimation

In the absence of noise, estimation is trivial, in that it amounts to successfully decoding the first window $\xi^{(i)}$, e.g., by BP; then all subsequent stream entries can be plainly retrieved by solving a redundant consistent set of linear equations $\boldsymbol{y}^{(i+1)} = \boldsymbol{y}^{(i)} + (x_{i+n} - x_i)\boldsymbol{a}_1^{(i)}$ where the only unknown is x_{i+n}. For noisy measurements, however, this approach is not a valid option due to error propagation: it is no longer true that $\hat{\boldsymbol{x}}^{(i)} = \boldsymbol{x}^{(i)}$, so computing x_{i+n} via (19) leads to accumulated errors and poor performance.

For *recursive estimation* we seek to find an estimate $\hat{\boldsymbol{x}}^{(i+1)} = \left[\hat{x}_0^{(i+1)} \dots \hat{x}_{n-1}^{(i+1)}\right]$ leveraging the estimate $\hat{\boldsymbol{x}}^{(i)} = \left[\hat{x}_0^{(i)} \dots \hat{x}_{n-1}^{(i)}\right]$ and using LASSO

$$\hat{x}^{(i+1)} = \arg\min_x \quad \|\boldsymbol{A}^{(i+1)}\boldsymbol{x} - \boldsymbol{y}^{(i+1)}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \quad .$$

In iterative schemes for convex optimization, convergence speed depends on the distance of the starting point to the optimal solution [26]. In order to accelerate convergence, we leverage the overlap between windows and set the starting point as:

$$\hat{\boldsymbol{x}}_{[0]}^{(i)} = \begin{bmatrix} \hat{x}_1^{(i-1)} & \hat{x}_2^{(i-1)} & \dots & \hat{x}_{n-1}^{(i-1)} & * \end{bmatrix}^\top,$$

where $\hat{x}_j^{(i-1)}$, for $j = 1, \dots, n-1$, is the portion of the optimal solution based on the previous window; we set $\hat{x}_j^{(i-1)}, j = 0, 1, \dots, n-1$ to be the estimate of the $(j+1)$-th entry of the previous window, i.e., of x_{i-1+j}. The last entry $\hat{x}_{n-1}^{(i)}$ (denoted by "*" above) can be selected using prior information on the data source; for example, for randomly generated sequence, the maximum likelihood estimate $\mathbb{E}_{\boldsymbol{x}^{(i-1)}}[x_{i+n-1}]$ may be a reasonable option, or we can simply set $\hat{x}_{n-1}^{(i)} = 0$, given that the sequence is assumed sparse. By choosing the starting point as such, the expected number of iterations for convergence is reduced (cf. Section V for a quantitative analysis).

In the general case where the signal is sparsely-representable in an orthonormal basis, one can leverage the recursive update for $\boldsymbol{\alpha}^{(i+1)}$ (based on $\boldsymbol{\alpha}^{(i)}$) so as to acquire an initial estimate for warm start in recursive estimation, e.g., $\mathbb{E}\left[\boldsymbol{\alpha}^{(i+1)}|\boldsymbol{\alpha}^{(i)}\right]$.

D. Averaging LASSO Estimates

One way to enhance estimation accuracy, i.e., to reduce estimation error variance, is to average the estimates obtained from successive windows. In particular, for the i^{th} entry of the streaming signal, x_i, we may obtain an estimate by averaging[1] the values corresponding to x_i obtained from *all* windows that contain the value, i.e., $\{\hat{\boldsymbol{x}}^{(j)}\}_{j=i-n+1,\dots,i}$:

$$\bar{x}_i := \frac{1}{n}\sum_{j=i-n+1}^{i} \hat{x}_{i-j}^{(j)}. \tag{24}$$

By Jensen's inequality, we get:

$$\frac{1}{n}\sum_{j=i-n+1}^{i}\left(\hat{x}_{i-j}^{(j)} - x_i\right)^2 \geq \left(\frac{1}{n}\sum_{j=i-n+1}^{i}\left(\hat{x}_{i-j}^{(j)} - x_i\right)\right)^2 = \left(\bar{x}_i - x_i\right)^2,$$

which implies that averaging may only decrease the reconstruction error–defined in the ℓ_2-sense. In the following, we analyze the expected ℓ_2-norm of the reconstruction error $\left(\bar{x}_i - x_i\right)^2$. We first present an important lemma establishing independence of estimates corresponding to different windows.

Lemma III.3 (Independence of estimates)**.** *Let $\boldsymbol{y}^{(i)} = \boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{w}^{(i)}$, $i = 0, 1, \cdots$, and $\{\boldsymbol{w}^{(i)}\}$ be independent, zero mean random vectors. The estimates $\{\hat{\boldsymbol{x}}^{(i)}\}$ obtained by LASSO,*

$$\hat{\boldsymbol{x}}^{(i)} := \operatorname*{argmin}_{\boldsymbol{x}} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}^{(i)}|| + \lambda||\boldsymbol{x}||_1$$

are independent (conditioned on the input stream $\boldsymbol{x} := \{x_i\}$[2]).

Proof. The objective function of LASSO

$$(\boldsymbol{x}, \boldsymbol{w}) \mapsto f(\boldsymbol{x}, \boldsymbol{w}) := \|\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$$

is jointly continuous in $(\boldsymbol{x}, \boldsymbol{w})$, and the mapping obtained by minimizing over $\boldsymbol{x}$

$$\boldsymbol{w} \mapsto \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{w}) := g(\boldsymbol{w})$$

is continuous, hence Borel measurable. Thus, the definition of independence and the fact that $\boldsymbol{w}^{(i)}, \boldsymbol{w}^{(j)}$ are independent for $i \neq j$ concludes the proof. □

The expected ℓ_2-norm of the reconstruction error satisfies:

$$\mathbb{E}_x\left[(\bar{x}_i - x_i)^2\right] = \mathbb{E}_x\left[\left(\frac{1}{n}\sum_{j=i-n+1}^{i}\hat{x}_{i-j}^{(j)} - x_i\right)^2\right]$$
$$= \left(\mathbb{E}_x\left[\hat{x}_0^{(i)}\right] - x_i\right)^2 + \frac{1}{n}\mathbb{E}_x\left[\left(\hat{x}_0^{(i)} - \mathbb{E}_x\left[\hat{x}_0^{(i)}\right]\right)^2\right],$$

where we have used $\operatorname{Cov}\left[\hat{x}_{i-j}^{(j)}, \hat{x}_{i-k}^{(k)}\right] = 0$ for $j \neq k$, $j, k \in \{i-n+1, \dots, i\}$ which follows from independence. The resulting equality is the so called *bias-variance* decomposition of the estimator. Note that as the window length is increased, the second term goes to zero and the reconstruction error asymptotically converges to the square of the LASSO bias[3].

[1]For notational simplicity, we consider the case $i \geq n-1$, whence each entry i is included in exactly n overlapping windows. The case $i < n-1$ can be handled analogously by considering $i+1$ estimates instead.

[2]This accounts for the general case of a random input source $\boldsymbol{x}$, where noise $\{\boldsymbol{w}^{(i)}\}$ is independent of $\boldsymbol{x}$

[3]LASSO estimator is *biased* as a mapping from $\mathbb{R}^m \to \mathbb{R}^n$ with $m < n$.

We have seen that averaging helps improve estimation accuracy. However, averaging, alone, is not enough for good performance, cf. Sec. VI, since the error variance is affected by the LASSO bias, even for large values of window size n. In the sequel, we propose a non-linear scheme for combining estimates from multiple windows which can overcome this limitation.

E. The Proposed Algorithm

In the previous section, we pointed out that leveraging the overlaps between windows–through averaging LASSO estimates–cannot yield an unbiased estimator, and the error variance does not go to 0 for large values of window size n. The limitation is the indeterminacy in the support of the signal–if the signal support is known, then applying least squares estimation (LSE) to an *overdetermined* linear systems yields an unbiased estimator. In consequence, it is vital to address *support detection*.

We propose a two-step estimation procedure for recovering the data stream: At first, we obtain the LASSO estimates $\{\hat{\boldsymbol{x}}^{(i)}\}$ which are fed into a *de-biasing* algorithm. For de-biasing, we estimate the signal support and then perform LSE on the support set in order to obtain estimates $\tilde{\boldsymbol{x}}^{(i)}$. The estimates obtained over successive windows are subsequently averaged. The block diagram of the method and the pseudocode for the algorithm can be seen in Figure 3 and Algorithm 1, respectively. In step 8, we show a recursive estimation of averages, applicable to an online implementation. In the next section, we present an efficient method for support detection with provable performance guarantees.

Algorithm 1 Recursive Compressed Sensing

Input: $\boldsymbol{A}^{(0)} \in \mathbb{R}^{m\times n}$, $\{x_i\}_{i=0,1,\ldots}$, $\lambda \geq 0$
Output: estimate $\{\bar{x}_i\}_{i=0,1,\ldots}$.

1: initialize signal estimate: $\{\bar{x}\} \leftarrow \{0\}$
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: $\quad \boldsymbol{x}^{(i)} \leftarrow [x_i\ x_{i+1}\ \ldots\ x_{i+n-1}]^\top$
4: $\quad \boldsymbol{y}^{(i)} \leftarrow \boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{w}^{(i)}$ ▷ encoding
5: $\quad \hat{\boldsymbol{x}}^{(i)} \leftarrow \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} \|\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{y}^{(i)}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$ ▷ LASSO
6: $\quad I \leftarrow \operatorname{supp}\left(\hat{\boldsymbol{x}}^{(i)}\right)$ ▷ support estimation
7: $\quad \tilde{\boldsymbol{x}}^{(i)} \leftarrow \underset{\substack{\boldsymbol{x}\in\mathbb{R}^n \\ \boldsymbol{x}_{\mathcal{I}^c}=\boldsymbol{0}}}{\operatorname{argmin}} \|\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{y}^{(i)}\|_2^2$ ▷ LSE
8: $\quad \bar{x}_{i+j} \leftarrow \left((k_i(j)-1)\bar{x}_{i+j} + \tilde{x}_j^{(i)}\right)/k_i(j)$ for $j = 0, \ldots, n-1$ where $k_i(j) = \min\{i+1, n-j\}$ ▷ update average estimates
9: $\quad \boldsymbol{A}^{(i)} \leftarrow \boldsymbol{A}^{(i-1)}\mathbf{P}$ ▷ for recursive sampling
10: **end for**

F. Voting strategy for support detection

Recall the application of LASSO to signal support estimation covered in Section II. In this section, we introduce a method utilizing supports estimated over successive windows for robust support detection even in high measurement noise. At first step, LASSO is used for obtaining estimate $\hat{\boldsymbol{x}}^{(i)}$, which is then used as input to a voting algorithm for estimating the non-zero positions. Then, ordinary least squares are applied to the overdetermined system obtained by extracting the columns of the sampling matrix corresponding to the support. The benefit is that, since LSE is an unbiased estimator, averaging estimates obtained over successive windows may eliminate the bias, and so it is possible to converge to *true* values as the window length increases.

In detail, the two-step algorithm with voting entails solving LASSO:

$$\hat{\boldsymbol{x}}^{(i)} = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} \left(\|\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{y}^{(i)}\|_2^2 + \lambda\|\boldsymbol{x}\|_1\right),$$

then identifying the indices having magnitude larger than some predetermined constant $\xi_1 > 0$, in order to estimate the support of window $\boldsymbol{x}^{(i)}$ by:

$$\hat{I}_i \coloneqq \left\{j : |\hat{\boldsymbol{x}}_j^{(i)}| \geq \xi_1\right\}. \tag{25}$$

The entries of this set are given a *vote*; the total number of votes determines whether a given entry is zero or not. Formally, we define the sequence containing the cumulative votes as $\{v_i\}$ and the number of times an index i is used in LSE as $\{l_i\}$. At the beginning of the algorithm $\{v_i\}$ and $\{l_i\}$ are all set to zero. For each window, we add votes on the positions that are in the set $\hat{I}i$ as $v_{\hat{I}_i+i} \leftarrow v_{\hat{I}_i+i} + 1$ (where the subscript $\hat{I}_i + i$ is used to translate the indices within the window to global indices on the streaming data). By applying threshold $\xi_2 \in \mathbb{Z}^+$ on the number of votes $\{v_i\}$, we get indices that have been voted sufficiently many times to be accepted as non-zeros and store them in:

$$R_i = \{j : v_{j+i} \geq \xi_2,\ j = 0, \ldots, n-1\}. \tag{26}$$

Note that the threshold $\xi_2 \in \{1, \cdots, n\}$ is equal to the *delay* in obtaining estimates. This can be chosen such that $|R_i| < m$, hence yielding an *overdetermined* system for the LSE. Subsequently, we solve the overdetermined least squares problem based on these indices in R_i,

$$\tilde{\boldsymbol{x}}^{(i)} = \underset{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{x}_{R_i^c}=0}{\operatorname{argmin}} \|\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{y}^{(i)}\|_2^2. \tag{27}$$

This problem can be solved in closed form, $\tilde{\boldsymbol{x}}_{R_i}^{(i)} = \left(\boldsymbol{A}_{R_i}^{(i)\top}\boldsymbol{A}_{R_i}^{(i)}\right)^{-1}\boldsymbol{A}_{R_i}^{(i)\top}\boldsymbol{y}^{(i)}$, where $\tilde{\boldsymbol{x}}_{R_i}^{(i)}$ is the vector obtained by extracting elements indexed by R_i, and $\boldsymbol{A}_{R_i}^{(i)}$ is the matrix obtained by extracting columns of $\boldsymbol{A}^{(i)}$ indexed by R_i. Subsequently, we increment the number of recoveries for the entries used in LSE procedure as $l_{R_i+i} \leftarrow l_{R_i+i} + 1$, and the average estimates are updated based on the recursive formula $\bar{x}_{i+j} \leftarrow \frac{l_{i+j}-1}{l_{i+j}}\bar{x}_{i+j} + \frac{1}{l_{i+j}}\tilde{x}_j$, for $j \in R_i$.

IV. Extensions

In this section we present various extensions to the algorithm.

A. Sliding window with step size τ

Consider a generalization in which sensing is performed via recurring windowing with a step size $0 < \tau \leq n$, i.e.,

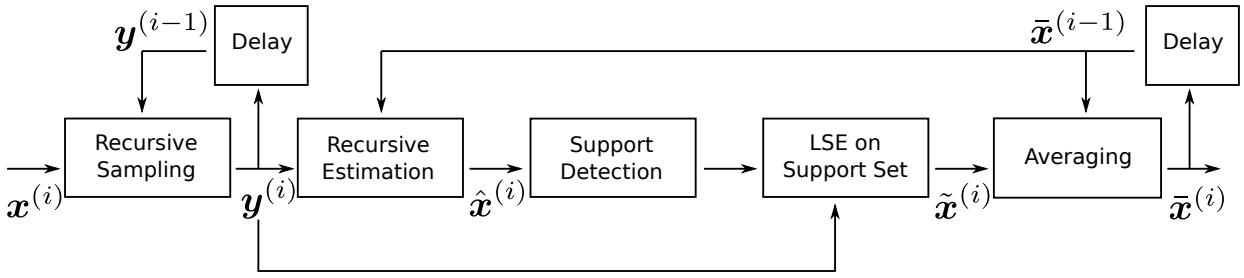


Fig. 3: Architecture of RCS.

$\boldsymbol{x}^{(i)} := \begin{bmatrix} x_{i\tau} & x_{i\tau+1} & \dots & x_{i\tau+n-1} \end{bmatrix}^\top$.

We let η_i denote the *sampling efficiency*, that is the ratio of the total number of samples taken until time $n+i$ to the number of retrieved entries, $n+i$. For one window, sampling efficiency is m/n. By the end of i^{th} window, we have recovered $n + (i-1)\tau$ elements while having sensed im many samples. The asymptotic sampling efficiency is:

$$\eta := \lim_{i\to\infty} \eta_i = \lim_{i\to\infty} \frac{im}{n+(i-1)\tau} = \frac{m}{\tau}.$$

The alternative is to encode using a rank-τ update (i.e., by recursively sampling using the matrix obtained by circularly shifting the sensing matrix τ times, $\boldsymbol{A}^{(i+1)} = \boldsymbol{A}^{(i)}\boldsymbol{P}^\tau$). In this scheme, for each window we need to store τ scalar parameters; for instance, this can be accomplished by a least-squares fit of the difference $\boldsymbol{y}^{(i+1)} - \boldsymbol{y}^{(i)}$ in the linear span of the first τ columns of $\boldsymbol{A}^{(i)}$ (cf. (20)). The asymptotic sampling efficiency becomes[4]:

$$\eta = \lim_{i\to\infty} \frac{m+(i-1)\tau}{n+(i-1)\tau} = 1.$$

In the latter case, the recursive sampling approach is asymptotically equivalent to taking one sample for each time instance. Note, however, that the benefit of such an approach lies in noise suppression. By taking overlapping windows each element is sensed at minimum $\lfloor n/\tau \rfloor$ many times, hence collaborative decoding using multiple estimates can be used to increase estimation accuracy.

B. Alternative support detection

The algorithm explained in Sec. III-F selects indices to be voted by thresholding the LASSO estimate as in (25). An alternative approach is by leveraging the estimates obtained so far: since we have prior knowledge about the signal at i^{th} window $\tilde{\boldsymbol{x}}^{(i)}_{[0]}$ from $(i-1)^{th}$ window, $\tilde{\boldsymbol{x}}^{(i-1)}$, we can annihilate the sampled signal as:

$$\tilde{\boldsymbol{y}}^{(i)} := \boldsymbol{y}^{(i)} - \boldsymbol{A}^{(i)}\tilde{\boldsymbol{x}}^{(i)}_{[0]}.$$

If the recovery of the previous window was perfect, $\tilde{\boldsymbol{y}}^{(i)}$ would be equal to $\boldsymbol{a}^{(i)}_n \boldsymbol{x}_{i+n-1} + \boldsymbol{w}^{(i)}$ and thus $\boldsymbol{x}_{i+n-1}$ can be estimated by LSE as $\boldsymbol{x}_{i+n-1} = \boldsymbol{a}^{(i)\top}_n \tilde{\boldsymbol{y}}^{(i)}$. However, since the previous window will have estimation errors, this does not hold. In such case, we can again use LASSO to find the estimator for the error between the true signal and estimate as:

$$\hat{\boldsymbol{x}}^{(i)} = \arg\min_{\boldsymbol{x}} \left(\|A^{(i)}\boldsymbol{x} - \tilde{\boldsymbol{y}}^{(i)}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \right)$$

and place votes on the $\xi_3 \in \mathbb{Z}^+$ indices of highest magnitudes, i.e.,

$$S^{(i)}_t = \left\{ j : |\hat{\boldsymbol{x}}^{(i)}_j| \geq z_{\xi_3} \right\}. \tag{28}$$

instead of (25). The rest of the estimation method remains the same. Since the noise is i.i.d., the expected number of votes a non-support position collects is less than ξ_3. Thus the threshold ξ_2 in (26) needs to satisfy $\xi_3 \leq \xi_2 \leq n$ in order to eliminate false positives.

Last, in the spirit of *recursive least squares* (RLS) [27], we consider joint identification over multiple windows with *exponential forgetting*. Let T be the *horizon*, i.e., the number of past windows considered in the estimation of the current one. Also, let $\rho \in [0,1)$. For the $i-$th window[5] we solve:

$$\boldsymbol{x}^* := \underset{\boldsymbol{x}\in\mathbb{R}^{n+T}}{\operatorname{argmin}} \sum_{j=i-T}^{i} \rho^{j-i} \left(||\boldsymbol{A}^{(j)}\boldsymbol{x}^{(j)} - \boldsymbol{y}^{(j)}||_2^2 + \lambda||\boldsymbol{x}^{(j)}||_1] \right), \tag{29}$$

where the decision vector $\boldsymbol{x}$ corresponds to $[x_{i-T}, \cdots, x_{i+n-1}]$, and we set $\hat{\boldsymbol{x}}^{(i)} := [\boldsymbol{x}^*_T, \cdots, \boldsymbol{x}^*_{n+T}$. It is interesting to point out that this optimization problem can be put into standard LASSO form by weighting the entries of the decision vector at a pre-processing step, so standard numerical schemes can be applied. Note that the computational complexity is increasing with T and, unlike traditional RLS, T has to be finite.

C. Expected Signal Sparsity

We have considered, so far, the case that each window $\boldsymbol{x}^{(i)}$ is κ-sparse. However, the most general case is when the data stream is κ-sparse *on average*, in the sense that:

$$\bar{\kappa} := \limsup_N \frac{1}{N} \sum_{i=0}^{N} 1_{x_i \neq 0} \leq \kappa.$$

In such case, one can simply design RCS based on some value $\kappa \geq \bar{\kappa}$, and leverage Theorem II.1 to incorporate the error due to *model-mismatch* in the analysis (cf. Theorem V.1). For both analysis and experiments we adopt a random model, in which

[4]Note that when $\tau \geq m$, recording samples $\{\boldsymbol{y}^{(i)}\}$ directly (as opposed to storing τ parameters) yields better efficiency $\eta = \frac{m}{\tau} \leq 1$.

[5]We consider the case $i \geq T$, and $\tau = 1$ for notational simplicity.

each entry of the data stream is generated i.i.d. according to:

$$f_{X_i}(x) = \begin{cases} (1-p)\delta(x) + \frac{1}{2p} & \text{if } x \in [-A, A] \\ 0 & \text{o.w.} \end{cases} \tag{30}$$

where $p \in (0, 1]$. This is the density function of a random variable that is 0 with probability $1-p$ and sampled uniformly over the interval $[-A, A]$ otherwise[6]. The average sparsity of the stream is $\bar{\kappa} = p$.

We can calculate the mean error due to model-mismatch by:

$$\mathbb{E}\left[\|\boldsymbol{X} - \boldsymbol{X}_\kappa\|_1\right] = A \sum_{k=\kappa+1}^{n} \binom{n}{k} p^k (1-p)^{n-k} \sum_{i=\kappa+1}^{k} \left(1 - \frac{i}{k+1}\right),$$

where $\boldsymbol{X}$ denotes an n-dimensional random vector with entries generated i.i.d. from (30), and $\boldsymbol{X}_k$ is obtained from $\boldsymbol{X}$ by setting all but its κ largest entries equal to zero.

The result is a function of the window length, n, the sparsity κ used in designing sensing matrices (e.g., we can take mean sparsity $\kappa = pn$), and the probability of an element being nonzero, p. In place of the (rather lengthy, yet elementary) algebraic calculations we illustrate error due to model-mismatch in Fig. 4. We point out that we can analytically establish boundedness for all values of p, n, so our analysis in Sec. V carries over unaltered. The analysis of other distributions on the magnitudes of non-zero entries can be carried out in a similar way.

V. Analysis

In this section we analyze the estimation error variance and computational complexity of the proposed method.

A. Estimation Error Variance

Given $\{x_i\}$ we give a bound on the normalized error variance of each window defined as:

$$\mathrm{NE}(i) \coloneqq \mathbb{E}\left[\frac{\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2}{\|\boldsymbol{x}^{(i)}\|_2}\right].$$

Note that for an ergodic data source this index (its inverse expressed in log-scale) corresponds to average Signal to Residual Ratio (SRR).

Theorem V.1 (Normalized Error of RCS)**.** *Under the assumptions of Theorem II.2 and given $\boldsymbol{A}^{(0)}$ satisfying RIP with δ_κ, for $\{x_i\}_{i=0,1,\ldots}$ satisfying $\|\boldsymbol{x}^{(i)}\|_0 \geq \Omega(\kappa)$, NE(i) satisfies:*

$$\begin{aligned} \textit{NE(i)} \leq & P^n \cdot c_1 \frac{1}{\sqrt{n \log n}} \\ & + (1 - P^n)) \left(c_2 + c_3 \frac{\sqrt{m}}{\sqrt{\kappa \log n}}\right), \end{aligned}$$

where c_1, c_2 and c_3 are constants, and

$$P^n \geq \left(1 - \frac{2}{n\sqrt{2\pi \log n}} - \frac{2\kappa}{n^2} - O\left(\frac{1}{n^{2\log 2}}\right)\right)^{2n-1}$$

[6]Note that the case $p = 0$ is trivially excluded since then the data stream is an all-zero sequence.

Proof. Defining the event $\mathrm{S}_{2n-1} \coloneqq \{$support is detected correctly on $2n-1$ consecutive windows$\}$[7], we have the following equality for NE given $\{x_i\}_{i=0,1,\ldots}$:

$$\begin{aligned} \mathrm{NE}(i) = & P(\mathrm{S}_{2n-1}) \cdot \mathbb{E}_{x, S_{2n-1}} \left[\frac{\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2}{\|\boldsymbol{x}^{(i)}\|_2}\right] \\ & + (1 - P(\mathrm{S}_{2n-1})) \cdot \mathbb{E}_{x, S^c_{2n-1}} \left[\frac{\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2}{\|\boldsymbol{x}^{(i)}\|_2}\right], \end{aligned}$$

where, dropping the subscript $2n-1$, and using S as a shorthand notation for S_{2n-1}, we have:

$$P(\mathrm{S}) \geq \left(1 - \frac{2}{n\sqrt{2\pi \log n}} - \frac{2\kappa}{n^2} - O\left(\frac{1}{n^{2\log 2}}\right)\right)^{2n-1},$$

by Theorem II.2.

In S, by LSE we get:

$$\begin{aligned} \mathbb{E}_{x,S}\left[\frac{\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2}{\|\boldsymbol{x}^{(i)}\|_2}\right] &\leq \frac{1}{\|\boldsymbol{x}^{(i)}\|_2} \sqrt{\mathbb{E}_{x,S}\left[\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2^2\right]} \\ &\overset{(a)}{\leq} \frac{\sigma\sqrt{\kappa}}{\|\boldsymbol{x}^{(i)}\|_2 \sqrt{n(1-\delta_\kappa)}}, \end{aligned}$$

where (a) follows from

$$\begin{aligned} &\mathbb{E}_{x,S}\left[\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2^2\right] = \mathbb{E}_{x,S}\left[\sum_{j \in I} \left(\bar{x}_{i+j} - x_{i+j}\right)^2\right] \\ &= \mathbb{E}_{x,S}\left[\sum_{j \in I} \left(\frac{1}{n} \sum_{t=0}^{n-1} \tilde{x}_{j+t}^{(i+j-t)} - x_{i+j}\right)^2\right] \\ &= \sum_{j \in I} \sum_{t,r=0}^{n-1} \mathbb{E}_{x,S}\left[\frac{\left(\tilde{x}_{j+t}^{(i+j-t)} - x_{i+j}\right)\left(\tilde{x}_{j+r}^{(i+j-r)} - x_{i+j}\right)}{n^2}\right] \\ &= \frac{1}{n^2} \sum_{j \in I} \sum_{t=0}^{n-1} \mathbb{E}_{x,S}\left[\left(\tilde{x}_{j+t}^{(i+j-t)} - x_{i+j}\right)^2\right] \\ &\overset{(b)}{\leq} \frac{1}{n^2} \sum_{j \in I} \sum_{t=0}^{n-1} \frac{\sigma^2}{1-\delta_\kappa} \leq \frac{\kappa\sigma^2}{n(1-\delta_\kappa)} \end{aligned}$$

where $I = \mathrm{supp}(\hat{\boldsymbol{x}}^{(i)})$, is also equal to $\mathrm{supp}(\boldsymbol{x}^{(i)})$ given S, and (b) follows since the covariance matrix of LSE is $\sigma^2(\boldsymbol{A}_I^T \boldsymbol{A}_I)^{-1}$ and by RIP we have all of the eigenvalues of $\boldsymbol{A}_I^\top \boldsymbol{A}_I$ greater than $(1-\delta_\kappa)$ since $(1-\delta_\kappa)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2$ for all $\boldsymbol{x}$ κ-sparse.

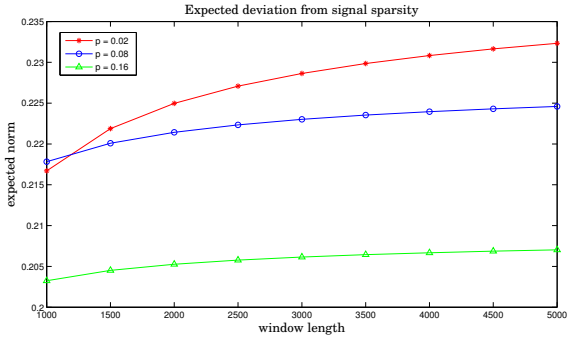
To bound the estimation error in S^c, note that independent of the selected support, by triangle inequality, we have:

$$\begin{aligned} \|\boldsymbol{A}^{(i)}\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{y}^{(i)}\|_2 &\overset{(a)}{\leq} \|\boldsymbol{y}^{(i)}\|_2 \\ &\leq \|\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)}\|_2 + \|\boldsymbol{w}^{(i)}\|_2 \\ &\leq (1+\delta_\kappa)\|\boldsymbol{x}^{(i)}\|_2 + \|\boldsymbol{w}^{(i)}\|_2, \end{aligned}$$

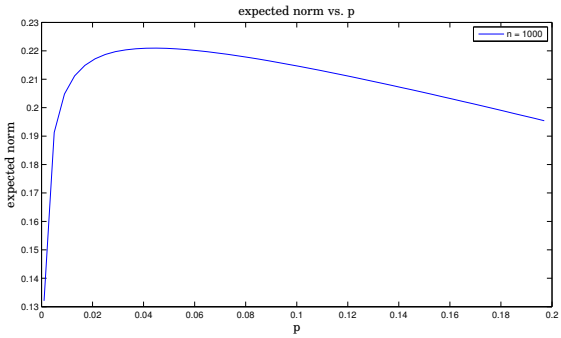
and

$$\begin{aligned} \|\boldsymbol{y}^{(i)}\|_2 \geq \|\boldsymbol{A}^{(i)}\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{y}^{(i)}\|_2 &\geq \|\boldsymbol{A}^{(i)}\tilde{\boldsymbol{x}}^{(i)}\|_2 - \|\boldsymbol{y}^{(i)}\|_2 \\ &\geq (1-\delta_\kappa)\|\tilde{\boldsymbol{x}}^{(i)}\|_2 - \|\boldsymbol{y}^{(i)}\|_2, \end{aligned}$$

[7]Note that the definition of the "success" set is very conservative.



(a) $\mathbb{E}\left[\|\boldsymbol{x}-\boldsymbol{x}_\kappa\|_1\right]$ vs. window length n;$\kappa = np$.



(b) $\mathbb{E}\left[\|\boldsymbol{x}-\boldsymbol{x}_\kappa\|_1\right]$ vs. probability of non-zero p; $\kappa = np$, $n = 1000$.

Fig. 4: Error due to model-mismatch: Expected deviation from signal sparsity $\mathbb{E}\left[\|\boldsymbol{x}-\boldsymbol{x}_\kappa\|_1\right]$.

where (a) follows since

$$\tilde{\boldsymbol{x}}^{(i)} = \underset{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{x}_{I^c}=0}{\operatorname{argmin}} \|\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)}\|_2.$$

From these two inequalities we have:

$$\begin{aligned}\|\tilde{\boldsymbol{x}}^{(i)}\|_2 &\leq \frac{2}{1-\delta_\kappa}\|\boldsymbol{y}^{(i)}\|_2 \\ &\leq \frac{2}{1-\delta_\kappa}\left((1+\delta_\kappa)\|\boldsymbol{x}^{(i)}\|_2 + \|\boldsymbol{w}^{(i)}\|_2\right).\end{aligned}$$

By applying triangle inequality once more we get:

$$\begin{aligned}\|\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2 &\leq \|\tilde{\boldsymbol{x}}^{(i)}\|_2 + \|\boldsymbol{x}^{(i)}\|_2 \\ &\leq \|\boldsymbol{x}^{(i)}\|_2\left(1 + \frac{2(1+\delta_\kappa)}{1-\delta_\kappa}\right) + \frac{2\|\boldsymbol{w}^{(i)}\|_2}{1-\delta_\kappa}.\end{aligned}$$

Thus in S^c we have:

$$\begin{aligned}&\mathbb{E}_{x,\mathrm{S}^c}\left[\frac{\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2}{\|\boldsymbol{x}^{(i)}\|_2}\right] \\ &= \frac{1}{\|\boldsymbol{x}^{(i)}\|_2}\mathbb{E}_{x,\mathrm{S}^c}\left[\|\bar{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2\right] \\ &\overset{(b)}{\leq} \frac{1}{\|\boldsymbol{x}^{(i)}\|_2}\mathbb{E}_{x,\mathrm{S}^c}\left[\|\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}\|_2\right] \\ &\leq \left(1 + \frac{2(1+\delta_\kappa)}{1-\delta_\kappa}\right) + \frac{2}{1-\delta_\kappa}\frac{\mathbb{E}\left[\|\boldsymbol{w}^{(i)}\|_2\right]}{\|\boldsymbol{x}^{(i)}\|_2},\end{aligned}$$

where (b) follows from Jensen's inequality.

We get the result by taking the expectation over $\{x_i\}_{i=0,1,\ldots}$ and noting by the assumptions of Theorem II.2 we have $\mathbb{E}\left[\|\boldsymbol{w}^{(i)}\|_2\right] \leq \sigma\sqrt{m}$, $|x_{i+j}| \geq 8\sigma\sqrt{2\log n}$ where $j \in \operatorname{supp}\left(\boldsymbol{x}^{(i)}\right)$ and $\|\boldsymbol{x}^{(i)}\|_0 \geq \Omega(\kappa)$. □

Corollary V.2. *For* sublinear *sparsity* $\kappa = O(n^{1-\epsilon})$*, non-zero data entries with magnitude* $\Omega(\sqrt{\log n})$*, and obtained samples* $m = O(\kappa \log n)$*, where* n *is the window length, the normalized error goes to 0 as* $n \to \infty$*.*

Proof. For $\kappa = O(n^{1-\epsilon})$ we have $P^n \geq \left(1 - O\left(\frac{1}{n\sqrt{\log n}}\right)\right)^{2n-1}$, and from the assumptions we have $c_3\frac{\sqrt{m}}{\sqrt{\kappa\log n}}$ constant. We get the result by noting P^n goes to 1 as the window length n goes to infinity. □

Remark 7 (Error of voting)**.** *Note that the exact same analysis applies directly to voting by invoking* stochastic dominance*: for any positive threshold* $\xi_1 \leq 8\sigma\sqrt{\log n}$*, correct detection occurs in a superset of* S_{2n-1} *(defined by requiring perfect detection in all windows, i.e.,* $\xi_1 = 0, \xi_2 = n$*).*

Remark 8 (Dynamic range[8])**.** *Note that in a real scenario, it may be implausible to increase the window length arbitrarily, because the* dynamic range *condition* $\min_{i\in I_{\boldsymbol{x}}} |x_i| > 8\sigma\sqrt{2\log n}$ *may be violated. This observation may serve to provide a means for selecting* n *(the good news being that the lower bound increases very slowly in window length, only as* $\sqrt{\log n}$*). In multiple simulations we have observed that this limitation is actually negligible:* n *can be selected way beyond this barrier without any compromise in increasing estimation accuracy.*

Last, we note that it is possible to carry out the exact same analysis for general step size τ; as expected, the upper bound on normalized error variance is increasing in τ, but we skip the details for length considerations.

B. Computational Complexity Analysis

In this section, we analyze the computational complexity of RCS. Let i be the window index, $\boldsymbol{A}^{(i)} \in \mathbb{R}^{m\times n}$ be the sampling matrix, and recall the extension on τ, the number of shifts between successive windows. By the end of i^{th} window, we have recovered $n + (i-1)\tau$ many entries. As discussed in Section III, the first window is sampled by $\boldsymbol{A}^{(0)}\boldsymbol{x}^{(0)}$; this requires $O(mn)$ basic operations (additions and multiplications). After the initial window, sampling of the window $\boldsymbol{x}^{(i)} = \begin{bmatrix} x_{i\tau} & x_{i\tau+1} & \cdots & x_{i\tau+n-1}\end{bmatrix}^\top$ is achieved by recursive sampling having rank-τ update with complexity $O(m\tau)$. Thus, by the end of i^{th} window, total complexity of sampling is $O(mn) + O(m\tau)i$. The *encoding complexity* is defined as the normalized complexity due to sampling over the number of *retrieved entries*:

$$C_e := \lim_{i\to+\infty} \frac{C_e(i)}{n + (i-1)\tau}, \tag{31}$$

where $C_e(i)$ denotes the total complexity of encoding all stream entries $0, 1, \ldots, i$. For recursive sampling $C_e = O(m)$ while for non-recursive we have $C_e = O(mn/\tau)$; note that

[8]The authors would like to thank Pr. Yoram Bresler for a fruitful comment on dynamic range.

by recursively sampling the input stream, the complexity is reduced by $\frac{n}{\tau}$.

The other contribution to computational complexity is due to the iterative solver, where the expected complexity can be calculated as the number of operations of a single iteration multiplied by the expected number of iterations for convergence. The latter is a function of the distance of the starting point to the optimal solution [26], which we bound in the case of using *recursive estimation*, as follows:

Lemma V.3. *Using* $\hat{\boldsymbol{x}}^{(i)}_{[0]} = \begin{bmatrix} x^{(i-1)}_{*\tau} & \dots & x^{(i-1)}_{*n-1} & \boldsymbol{0}^\top_\tau \end{bmatrix}^\top$ *as the starting point we have:*

$$\begin{aligned}\|\hat{\boldsymbol{x}}^{(i)}_{[0]} - \boldsymbol{x}^{(i)}_*\|_2 \leq\ & c_0\|\boldsymbol{x}^{(i-1)} - \boldsymbol{x}^{(i-1)}_\kappa\|_1/\sqrt{\kappa} \\ & + c_0\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}_\kappa\|_1/\sqrt{\kappa} \\ & + c_1\tilde{\sigma} + \| \begin{bmatrix} x^{(i)}_{n-\tau} \dots x^{(i)}_{n-1} \end{bmatrix} \|_2,\end{aligned}$$

where c_0 *and* c_1 *are constants.*

Proof. Defining:

$$\begin{aligned}\boldsymbol{e}'^{(i)} &\coloneqq \begin{bmatrix} x^{(i-1)}_{*\tau} \dots x^{(i-1)}_{*n-1} & \boldsymbol{0}^\top_\tau \end{bmatrix}^\top - \begin{bmatrix} x^{(i)}_{*0} \dots x^{(i)}_{*n-1} \end{bmatrix}^\top \\ \boldsymbol{e}^{(i)} &\coloneqq \boldsymbol{x}^{(i)}_* - \boldsymbol{x}^{(i)},\end{aligned}$$

we have

$$\boldsymbol{e}'^{(i)} = \begin{bmatrix} x^{(i-1)}_{*\tau} \dots x^{(i-1)}_{*n-1} & \boldsymbol{0}^\top_\tau \end{bmatrix}^\top - \boldsymbol{x}^{(i)} \quad + \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}_*.$$

Taking the norm and using triangle inequality yields:

$$\|\boldsymbol{e}'^{(i)}\|_2 \leq \|\boldsymbol{e}^{(i-1)}\|_2 + \|\boldsymbol{e}^{(i)}\|_2 + \| \begin{bmatrix} x^{(i)}_{n-\tau} \dots x^{(i)}_{n-1} \end{bmatrix} \|_2.$$

Using Theorem II.1 we get:

$$\begin{aligned}\|\boldsymbol{e}'^{(i)}\|_2 \leq\ & c_0\|\boldsymbol{x}^{(i-1)} - \boldsymbol{x}^{(i-1)}_\kappa\|_1/\sqrt{\kappa} \\ & + c_0\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}_\kappa\|_1/\sqrt{\kappa} \\ & + c_1\tilde{\sigma} + \| \begin{bmatrix} x^{(i)}_{n-\tau} \dots x^{(i)}_{n-1} \end{bmatrix} \|_2. \qquad (32)\end{aligned}$$

□

Exact computational complexity of each iteration depends on the algorithm. Minimally, iterative solver for LASSO requires multiplication of sampling matrix and the estimate at each iteration which requires $O(mn)$ operations. In an algorithm where cost function decays sublinearly (e.g., $1/t^2$), as in FISTA, the number of iterations, t, required for obtaining $\hat{\boldsymbol{x}}_{[t]}$ such that $G(\hat{\boldsymbol{x}}_{[t]}) - G(\boldsymbol{x}_*) \leq \epsilon$, where $\boldsymbol{x}_*$ is the optimal solution, is proportional to $\|\boldsymbol{x}_{[0]} - \boldsymbol{x}_*\|_2$ (e.g., $\|\boldsymbol{x}_{[0]} - \boldsymbol{x}_*\|_2/\sqrt{\epsilon}$) where $\boldsymbol{x}_{[0]}$ is the starting point of the algorithm [22]. From this bound, it is seen that average number of iterations is proportional to the Euclidean distance of the starting point of the algorithm from the optimal point.

Lemma V.4 (Expected number of iterations)**.** [9] *For the sequence* $\{x_i\}_{i=0,1,\dots}$ *where* $\|\boldsymbol{x}^{(i)}\|_0 \leq \kappa$ *with the positions of non-zeros chosen uniformly at random and* $\max_{j=0,\dots,n-1} |\boldsymbol{x}^{(i)}_j| = O\left(\sqrt{\log n}\right)$ *for all* i*, the expected number of iterations for convergence of algorithms where cost function decays as* $1/t^2$ *is* $O(\sqrt{(\kappa\tau \log n)/n})$ *for noiseless measurements and* $O(\sqrt{m})$ *for i.i.d. measurement noise.*

Proof. Since $\boldsymbol{x}^{(i)}$ is κ-sparse, the terms $\|\boldsymbol{x}^{(i-1)} - \boldsymbol{x}^{(i-1)}_\kappa\|_1$ and $\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}_\kappa\|_1$ vanish in (32). By $|x_i| = O\left(\sqrt{\log n}\right)$ and uniform distribution of non-zero elements we have $\mathbb{E}\left[\| \begin{bmatrix} x^{(i)}_{n-\tau} \dots x^{(i)}_{n-1} \end{bmatrix} \|_2\right] \leq \sqrt{(\kappa\tau \log n)/n}$.

With noisy measurements, the term $c_1\tilde{\sigma}$ is related to the noise level. Since noise has distribution $\boldsymbol{w}^{(\cdot)} \sim \mathcal{N}\left(0, \sigma^2\boldsymbol{I}\right)$, the squared norm of the noise $\|\boldsymbol{w}^{(i)}\|_2^2$ has chi-squared distribution with mean $\sigma^2 m$ and standard deviation $\sigma^2\sqrt{2m}$; probability of the squared norm exceeding its mean plus 2 standard deviations is small, hence we can pick $\tilde{\sigma}^2 = \sigma^2\left(m + 2\sqrt{2m}\right)$ [5] to satisfy the conditions of Theorem II.1. Using this result in (32), we get $O(\sqrt{(\kappa \log n\tau)/n}) + O(\sqrt{m})$, where the second term dominates since $\tau \leq n$ not to leave out any element of the signal and $m \sim O(\kappa \log n)$. Hence it is found that the expected number of iterations is $O(\sqrt{m})$ in the noisy case. □

The other source of complexity is the LSE in each iteration, which requires solving a linear $\kappa \times \kappa$ system that needs $O(\kappa^3)$ operations. Finally, averaging can be performed using $O(n/\tau)$ operations for each given entry. We define the *decoding complexity* as the normalized complexity due to estimation over the number of *retrieved entries*:

$$C_d \coloneqq \lim_{i \to +\infty} \frac{C_d(i)}{n + (i-1)\tau}, \qquad (33)$$

where $C_d(i)$ denotes the total complexity of decoding all stream entries $0, 1, \dots, i$. It follows that decoding complexity is equal to $C_d = O(\frac{m^{3/2}n + \kappa^3}{\tau})$, using recursive estimation. To conclude, the asymptotic total complexity (per retrieved stream entry),

$$C = C_e + C_d,$$

is dominated by LASSO and LSE (based on the facts that $m \geq 1, \frac{n}{\tau} \geq 1$), therefore:

$$C = O(\frac{m^{3/2}n + \kappa^3}{\tau}), \qquad (34)$$

In Table I we demonstrate the total complexity for various sparsity classes κ, based on the fundamental relation $m = O(\kappa \log \frac{n}{\kappa})$ [12]. Note that the computational complexity is decreasing in τ, while error variance is increasing in τ. This trade-off can be used for selecting window length n and step size τ based on desired estimation accuracy and real-time considerations.

VI. Simulation Results

The data used in the simulations are generated from the random model (30) [10] with $p = 0.05$. The measurement model is $\boldsymbol{y}^{(i)} = \boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{w}^{(i)}$ with $\boldsymbol{w}^{(i)} \sim \mathcal{N}\left(0, \sigma^2\boldsymbol{I}\right)$ where

[9] We note in passing that this bound on the expected number of iterations is actually conservative, and can be improved based on a homotopy analysis of warm-start [6], [8]; this is beyond the scope of the current paper.

[10] We also tested the case where the values of non-zero entries are generated i.i.d. from a Gaussian distribution; even though this model may violate the dynamic range assumption, cf. Rem. 8, the results are very similar.

TABLE I: Computational complexity per entry as function of window length n and step size τ for different sparsity classes.

κ	Computational Complexity
$O(1)$	$O\left(n(\log n)^{3/2}/\tau\right)$
$O(\log n)$	$O\left(n\left(\log n \cdot \log(n/\log n)\right)^{3/2}/\tau\right)$
$O(\sqrt{n})$	$O(n^{\frac{3}{2}}/\tau)$
$O(n)$	$O(n^3/\tau)$

$\sigma \in \mathbb{R}^+$, and the sampling matrix is $\boldsymbol{A}^{(0)} \in \mathbb{R}^{m\times n}$ where $m = 6pn$ and n is equal to the window length.

In the sequel, we test RCS as described in sections III-E, III-F. We have also experimented extensively on the extensions presented in Sec. IV-B, but do not present the results here because: a) the exponential-forgetting approach, alone, does not improve estimation accuracy while it incurs computation overhead, and b) the performance and run-time of generalized voting is no different than that of standard voting.

A. Runtime

We experimentally test the speed gain achieved by RCS by comparing the average time required to estimate a given window while using FISTA for solving LASSO. RCS is compared against so called 'naïve approach', where the sampling is done by matrix multiplication in each window and FISTA is started from all zero vector. The average time required to recover one window in each case is shown in Figure 5.

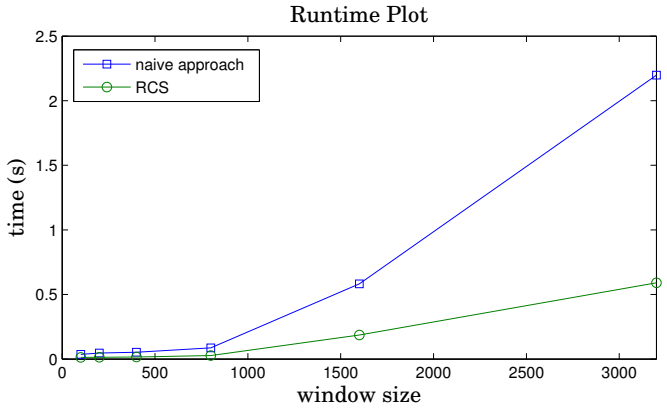


Fig. 5: Average processing time of RCS vs. traditional (non-recursive) CS over a single time window.

B. Support Estimation

We present the results of experiments on the support estimation using LASSO. In the measurements $\boldsymbol{x} \in \mathbb{R}^{6000}$, $\|\boldsymbol{x}\|_0 = 60$, $\boldsymbol{A} \in \mathbb{R}^{m\times 6000}$ is generated by i.i.d. Gaussian distribution with $A_{i,j} \sim \mathcal{N}(0, 1/m)$, and $\boldsymbol{w}$ has $\sigma = 0.1$. As suggested in Theorem II.2 for these parameters, LASSO is solved with $\lambda = 4\sigma\sqrt{2\log n}$, and the nonzero entries of $\boldsymbol{x}$ are chosen so that $\min_{i=1,2,\ldots,n} |x_i| \geq 3.34$ by sampling from $\mathcal{U}\left([-4.34, -3.34] \cup [3.34, 4.34]\right)$. In simulations, we vary the number of samples taken from the signal, m, and study the accuracy of support estimation by using

$$\text{true positive rate} = \frac{|\text{detected support} \cap \text{true support}|}{|\text{true support}|}$$

$$\text{false positive rate} = \frac{|\text{detected support} \backslash \text{true support}|}{n - |\text{true support}|},$$

where $|\cdot|$ denotes the cardinality of a set and $\backslash$ is the set difference operator.

The support is detected by taking the positions where the magnitude of the LASSO estimate is greater than threshold ξ_1 for values 0.01, 0.1, 1. Figure 6 shows the resulting curves, obtained by randomly generating the input signal 20 times for each m and averaging the results. It can be seen that the false positive rate can be reduced significantly by properly adjusting the threshold on the resulting LASSO estimates.

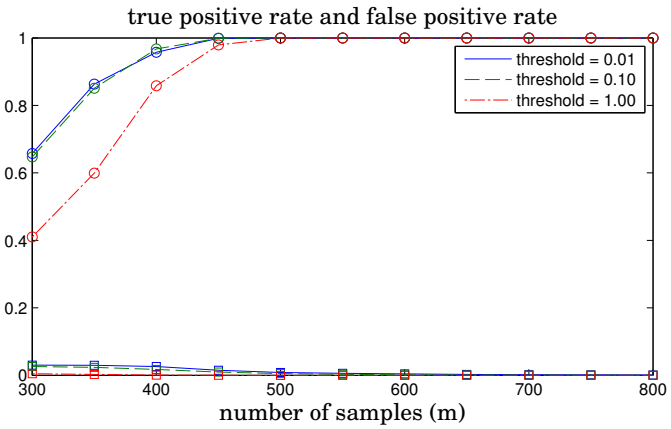


Fig. 6: Support set estimation using LASSO: for $n = 6000$, $\sigma = 0.1$, $\min|x_i| \geq 3.34$, threshold $\xi_1 = 0.01$, 0.10 and 1.00. Circles depict *true positive rate*, and squares depict *false positive rate*.

C. Reconstruction Error

As was discussed in Section III-F, LASSO can be used together with a voting strategy and least squares estimation to reduce error variance. Figure 7 shows the comparison of performance of a) averaged LASSO estimates, b) debiasing and averaging with voting strategy, and c) debiasing and averaging without voting. The figure is obtained by using fixed $\boldsymbol{x}$ (i.e., a single window) and taking multiple measurements (each being an m-dimensional vector) corrupted by i.i.d. Gaussian noise. It can be seen that the error does not decrease to zero for averaged estimate, which is due to LASSO being a biased estimator, cf. Section III, whereas for the proposed schemes it does.

Figure 8 shows the behavior of normalized error variance

$$\lim_{T\to\infty} \frac{\sum_{i=1}^{T}(\bar{x}_i - x_i)^2}{\sum_{i=1}^{T}(x_i)^2}$$

as the window length, n, increases. The signals are generated to be 5% sparse, m is chosen to be 5 times the expected window sparsity, and the measurement noise is $\boldsymbol{w}^{(i)} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$

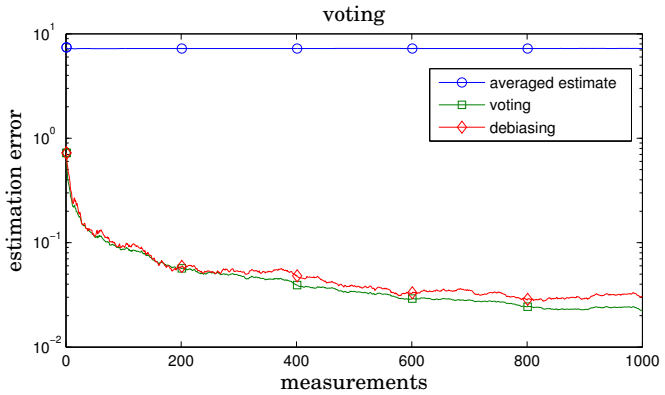


Fig. 7: Error plots for a) averaged estimates, b) voting strategy, and c) debiasing without voting.

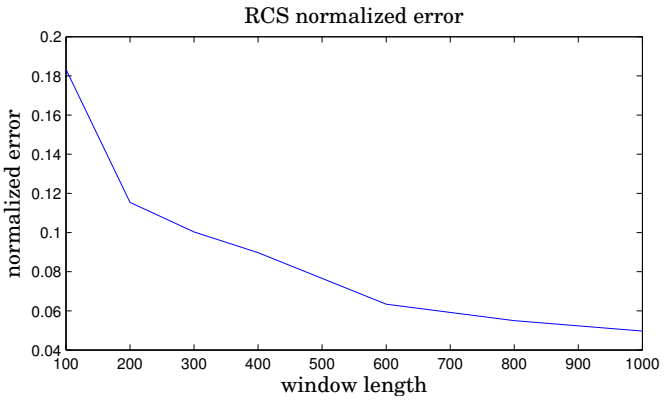


Fig. 8: Normalized error variance vs. window length for RCS on streaming data.

where $\sigma = 0.1$. The non-zero amplitudes of the signal are drawn from uniform distribution $\mathcal{U}([-2,-1]\cup[1,2])$ The figure shows that the normalized error variance decreases as the window length increases, which is in full agreement with our theoretical analysis.

VII. Conclusions and Future Work

We have proposed an efficient online method for compressively sampling data streams. The method uses a sliding window for data processing and entails recursive sampling and iterative recovery. By exploiting redundancy we achieve higher estimation accuracy as well as reduced run-time, which makes the algorithm suitable for an online implementation. Extensive experiments showcase the merits of our approach compared to traditional CS: a) at least 10x speed-up in run-time, and b) 2-3 orders of magnitude lower reconstruction error.

In ongoing work, we study accelerating the decoding procedure by deriving a fast LASSO solver directly applicable to RCS. We also seek to apply the derived scheme in practical applications such as burst detection in networks and channel estimation in wireless communications.

Acknowledgement

This work was supported in part by Qualcomm, San Diego, and ERC Advanced Investigators Grant, SPARSAM, no. 247006.

References

[1] N. Freris, O. Öçal, and M. Vetterli, "Compressed Sensing of Streaming data," in *51st Allerton Conference on Communication, Control and Computing*, 2013.

[2] E. Candès and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, March 2008.

[3] M. Vetterli, P. Marziliano, and T. Blu, "Sampling signals with finite rate of innovation," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1417–1428, 2002.

[4] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, 2006.

[5] E. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[6] P. Garrigues and L. El Ghaoui, "An homotopy algorithm for the lasso with online observations," in *Proc. NIPS*, 2008.

[7] P. Boufounos and M. Asif, "Compressive sampling for streaming signals with sparse frequency content," in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, 2010, pp. 1–6.

[8] M. S. Asif and J. Romberg, "Sparse recovery of streaming signals using L1-homotopy," Submitted to IEEE Transactions on Signal Processing, June 2013.

[9] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, Mar. 2009.

[10] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, dec. 2005.

[11] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2001, pp. 274–281.

[12] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[13] D. P. Bertsekas, *Convex Optimization Theory*. Athena Scientific, 2009.

[14] J. A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, pp. 2231–2242, 2004.

[15] R. Tibshirani, "Regression Shrinkage and Selection via the LASSO," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, pp. 267–288, 1996.

[16] S. Rangan, A. Fletcher, and V. Goyal, "Asymptotic Analysis of MAP Estimation via the Replica Method and Applications to Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 58, no. 3, pp. 1902–1923, Mar. 2012.

[17] E. Candès and Y. Plan, "Near-ideal model selection by ℓ_1 minimization," *The Annals of Statistics*, vol. 37, pp. 2145–2177, 2009.

[18] C.-H. Zhang and J. Huang, "The sparsity and bias of the LASSO selection in high-dimensional linear regression," *Annals of Statistics*, vol. 36, no. 4, pp. 1567–1594, 2008.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[20] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sep. 2012.

[21] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, 2013.

[22] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[23] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse Reconstruction by Separable Approximation," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2479–2493, 2009.

[24] M. Afonso, J. Bioucas-Dias, and M. A. T. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2345–2356, 2010.

[25] N. Freris and P. Patrinos, "PN-LASSO: A proximal Newton algorithm for Compressed Sensing," In preparation.

[26] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1995.

[27] P.R. Kumar and P. Varaiya, *Stochastic systems: estimation, identification and adaptive control*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1986.