Recursive $\ell_{1,\infty}$ Group lasso

Yilun Chen, Student Member, IEEE, and Alfred O. Hero, III, Fellow, IEEE

Abstract—We introduce a recursive adaptive group lasso algorithm for real-time penalized least squares prediction that produces a time sequence of optimal sparse predictor coefficient vectors. At each time index the proposed algorithm computes an exact update of the optimal $\ell_{1,\infty}$-penalized recursive least squares (RLS) predictor. Each update minimizes a convex but non-differentiable function optimization problem. We develop an online homotopy method to reduce the computational complexity. Numerical simulations demonstrate that the proposed algorithm outperforms the $\ell_1$ regularized RLS algorithm for a group sparse system identification problem and has lower implementation complexity than direct group lasso solvers.

Index Terms—RLS, group sparsity, mixed norm, homotopy, group lasso, system identification

I. INTRODUCTION

Recursive Least Squares (RLS) is a widely used method for adaptive filtering and prediction in signal processing and related fields. Its applications include: acoustic echo cancelation; wireless channel equalization; interference cancelation and data streaming predictors. In these applications a measurement stream is recursively fitted to a linear model, described by the coefficients of an FIR prediction filter, in such a way to minimize a weighted average of squared residual prediction errors. Compared to other adaptive filtering algorithms such as Least Mean Square (LMS) filters, RLS is popular because of its fast convergence and low steady-state error.

In many applications it is natural to constrain the predictor coefficients to be sparse. In such cases the adaptive FIR prediction filter is a sparse system: only a few of the impulse response coefficients are non-zero. Sparse systems can be divided into general sparse systems and group sparse systems [1], [2]. Unlike a general sparse system, whose impulse response can have arbitrary sparse structure, a group sparse system has impulse response composed of a few distinct clusters of non-zero coefficients. Examples of group sparse systems include specular multipath acoustic and wireless channels [3], [4] and compressive spectrum sensing of narrowband sources [5].

The exploitation of sparsity to improve prediction performance has attracted considerable interest. For general sparse systems, the $\ell_1$ norm has been recognized as an effective promoter of sparsity [6], [7]. In particular, $\ell_1$ regularized LMS [2], [8] and RLS [9], [10] algorithms have been proposed for sparsification of adaptive filters. For group sparse systems, mixed norms such as the $\ell_{1,2}$ norm and the $\ell_{1,\infty}$ norm have been applied to promote group-level sparsity in statistical regression [11]–[13], commonly referred to as the group lasso, and sparse signal recovery in signal processing and communications [1], [14]. In [2], the authors proposed a family of LMS filters with convex regularizers. As a specific scenario, they considered group sparse LMS filters with $\ell_{1,2}$-type regularizers and empirically demonstrated the superior performance of $\ell_{1,2}$-regularized LMS over the $\ell_1$-regularized LMS for identifying unknown group-sparse channels.

In this paper, we develop group sparse RLS algorithms for real-time system identification. Specifically, we consider RLS penalized by the $\ell_{1,\infty}$ norm to promote group sparsity, which we call the recursive $\ell_{1,\infty}$ group lasso. The algorithm is based on the homotopy approach to solving the lasso problem and is a nontrivial extension of [15]–[17] to group sparse structure. Both the $l_{1,2}$ and $l_{1,\infty}$ regularizer have been widely adopted to enforce group sparse structure in regression and other problems. While, as shown in [18], in some cases the $l_{1,2}$ is more effective at enforcing group sparsity, we adopt the $l_{1,\infty}$ norm due to its more favorable computational properties. As we show in this paper, the $l_{1,\infty}$ regularizer can be very efficiently implemented in the recursive setting of RLS. Our implementation exploits the piecewise linearity of the $l_{1,\infty}$ norm to obtain an exact solution to each iteration of the group sparsity-penalized RLS algorithm using the homotopy approach. Compared to other contemporary $l_{1,\infty}$ and $l_{1,2}$ group lasso solvers [12], [19], our simulation results demonstrate that our proposed methods attain equivalent or better performance but at lower computational cost for online group sparse RLS problems.

The paper is organized as follows. Section II formulates the problem. In Section III we develop the homotopy based algorithm to solve the recursive $l_{1,\infty}$ group lasso in an online recursive manner. Section IV provides numerical simulation results and Section V summarizes our principal conclusions. The proofs of theorems and some details of the proposed algorithm are provided in Appendix.

Notations: In the following, matrices and vectors are denoted by boldface upper case letters and boldface lower case letters, respectively; $(\cdot)^T$ denotes the transpose operator, and $\| \cdot \|_1$ and $\| \cdot \|_\infty$ denote the $\ell_1$ and $\ell_\infty$ norm of a vector, respectively; for a set $\mathcal{A}$, $|\mathcal{A}|$ denotes its cardinality and $\phi$ denotes the empty set; $x_{\mathcal{A}}$ denotes the sub-vector of $x$ from the index set $\mathcal{A}$ and $R_{AB}$ denotes the sub-matrix of $R$ formed from the row index set $A$ and column index set $B$.

II. PROBLEM FORMULATION

A. Recursive Least Squares

Let $w$ be a $p$-dimensional coefficient vector. Let $y$ be an $n$-dimensional vector comprised of observations $\{y_j\}_{j=1}^n$. Let
Recursive $l_{1,00}$ Group Lasso

Abstract

We introduce a recursive adaptive group lasso algorithm for real-time penalized least squares prediction that produces a time sequence of optimal sparse predictor coefficient vectors. At each time index the proposed algorithm computes an exact update of the optimal \('l_{1,1}\)-penalized recursive least squares (RLS) predictor. Each update minimizes a convex but nondifferentiable function optimization problem. We develop an online homotopy method to reduce the computational complexity. Numerical simulations demonstrate that the proposed algorithm outperforms the \('l_1\) regularized RLS algorithm for a group sparse system identification problem and has lower implementation complexity than direct group lasso solvers.
\{x_j\}_{j=1}^n be a sequence of \(p\)-dimensional predictor variables. In standard adaptive filtering terminology, \(y_j, x_j\) and \(w\) are the primary signal, the reference signal, and the adaptive filter weights. The RLS algorithm solves the following quadratic minimization problem recursively over time \(n = p,p+1,\ldots:\)

\[
\hat{w}_n = \arg\min_w \sum_{j=1}^n \gamma^{n-j}(y_j - w^T x_j)^2, \tag{1}
\]

where \(\gamma \in (0, 1]\) is the forgetting factor controlling the trade-off between transient and steady-state behaviors.

To serve as a template for the sparse RLS extensions described below we briefly review the RLS update algorithm. Define \(R_n\) and \(r_n\) as

\[
R_n = \sum_{j=1}^n \gamma^{n-j}x_j x_j^T \tag{2}
\]

and

\[
r_n = \sum_{j=1}^n \gamma^{n-j}x_j y_j. \tag{3}
\]

The solution \(\hat{w}_n\) to (1) can be then expressed as

\[
\hat{w}_n = R_n^{-1} r_n. \tag{4}
\]

The matrix \(R_n\) and the vector \(r_n\) are updated as

\[
R_n = \gamma R_{n-1} + x_n x_n^T,
\]

and

\[
r_n = \gamma r_{n-1} + x_n y_n^T.
\]

Applying the Sherman-Morrison-Woodbury formula [20],

\[
R_n^{-1} = \gamma^{-1} R_{n-1}^{-1} - \gamma^{-1} \alpha_n g_n g_n^T R_{n-1}^{-1}, \tag{5}
\]

where

\[
g_n = R_{n-1}^{-1} x_n \tag{6}
\]

and

\[
\alpha_n = \frac{1}{\gamma + x_n^T g_n}. \tag{7}
\]

Substituting (5) into (4), we obtain the weight update [21]

\[
\hat{w}_n = \hat{w}_{n-1} + \alpha_n g_n e_n, \tag{8}
\]

where

\[
e_n = y_n - \hat{w}_n^T x_n. \tag{9}
\]

Equations (5)-(9) define the RLS algorithm which has computational complexity of order \(O(p^2)\).

\[\text{Fig. 1. Examples of (a) a general sparse system and (b) a group-sparse system.}\]

\[\text{where } \{G_m\}_{m=1}^M \text{ is a group partition of the index set } G = \{1, \ldots, p\}, \text{i.e.,}\]

\[
\bigcup_{m=1}^M G_m = G, \quad G_m \cap G_{m'} = \emptyset \text{ if } m \neq m',
\]

and \(w_{G_m}\) is a sub-vector of \(w\) indexed by \(G_m\). The \(\ell_1,\infty\) norm is a mixed norm: it encourages correlation among coefficients inside each group via the \(\ell_\infty\) norm within each group and promotes sparsity across each group using the \(\ell_1\) norm. The mixed norm \(\|w\|_{1,\infty}\) is convex in \(w\) and reduces to \(\|w\|_1\) when each group contains only one coefficient, i.e., \(|G_1| = |G_2| = \cdots = |G_M| = 1\).

The \(\ell_1,\infty\) group lasso solves the following penalized least squares problem:

\[
\hat{w}_n = \arg\min_w \frac{1}{2} \sum_{j=1}^n \gamma^{n-j}(y_j - w^T x_j)^2 + \lambda \|w\|_{1,\infty}, \tag{10}
\]

where \(\lambda\) is a regularization parameter and as in standard RLS \(\gamma\) controls the trade-off between the convergence rate and steady-state performance. Eq. (10) is a convex problem and can be solved by standard convex optimizers or path tracing algorithms [12]. Direct solution of (10) has computational complexity of \(O(p^3)\).

\[\text{C. Recursive } \ell_1,\infty \text{ group lasso}\]

In this subsection we obtain a recursive solution for (10) that gives an update \(\hat{w}_n\) from \(\hat{w}_{n-1}\). The approach taken is a group-wise generalization of recent works [15], [16] that uses the homotopy approach to sequentially solve the lasso problem. Using the definitions (2) and (3), the problem (10) is equivalent to

\[
\hat{w}_n = \arg\min_w \frac{1}{2} w^T \gamma R_{n-1} w + \lambda \|w\|_{1,\infty} - w^T (\gamma R_{n-1} + x_n y_n) \tag{11}
\]

\[
= \arg\min_w \frac{1}{2} w^T (\gamma R_{n-1} + x_n x_n^T) w - w^T (\gamma R_{n-1} + x_n y_n) + \lambda \|w\|_{1,\infty}.
\]
Let \( f(\beta, \lambda) \) be the solution to the following parameterized problem

\[
f(\beta, \lambda) = \arg \min_w \frac{1}{2} w^T (\gamma R_n - 1 + \beta x_n x_n^T) w - w^T (\gamma r_n - 1 + \beta x_n y_n) + \lambda \|w\|_{1, \infty}
\]  

(12)

where \( \beta \) is a constant between 0 and 1. \( \hat{w}_n \) and \( \hat{w}_{n-1} \) of problem (11) can be expressed as

\[
\hat{w}_{n-1} = f(0, \gamma \lambda),
\]

and

\[
\hat{w}_n = f(1, \lambda).
\]

Our proposed method computes \( \hat{w}_n \) from \( \hat{w}_{n-1} \) in the following two steps:

**Step 1.** Fix \( \beta = 0 \) and calculate \( f(0, \lambda) \) from \( f(0, \gamma \lambda) \). This is accomplished by computing the regularization path between \( \gamma \lambda \) and \( \lambda \) using homotopy methods introduced for the non-recursive \( \ell_{1, \infty} \) group lasso. The solution path is piecewise linear and the algorithm is described in [12].

**Step 2.** Fix \( \lambda \) and calculate the solution path between \( f(0, \lambda) \) and \( f(1, \lambda) \). This is the key problem addressed in this paper.

To ease the notations we denote \( x_n \) and \( y_n \) by \( x \) and \( y \), respectively, and define the following variables:

\[
R(\beta) = \gamma R_n - 1 + \beta xx^T
\]

(13)

\[
r(\beta) = \gamma r_n - 1 + \beta xy.
\]

(14)

Problem (12) is then

\[
f(\beta, \lambda) = \arg \min_w \frac{1}{2} w^T R(\beta) w - w^T r(\beta) + \lambda \|w\|_{1, \infty}.
\]

(15)

In Section III we will show how to propagate \( f(0, \lambda) \) to \( f(1, \lambda) \) using the homotopy approach applied to (15).

### III. Online Homotopy Update

#### A. Set notation

We begin by introducing a series of set definitions. Figure 2 provides an example. We divide the entire group index set into \( \mathcal{P} \) and \( \mathcal{Q} \), respectively, where \( \mathcal{P} \) contains active groups and \( \mathcal{Q} \) is its complement. For each active group \( m \in \mathcal{P} \), we partition the group into two parts: the maximal values, with indices \( \mathcal{A}_m \), and the rest of the values, with indices \( \mathcal{B}_m \):

\[
\mathcal{A}_m = \arg \max_{i \in \mathcal{G}_m} |w_i|, \quad m \in \mathcal{P},
\]

and

\[
\mathcal{B}_m = \mathcal{G}_m - \mathcal{A}_m.
\]

The set \( \mathcal{A} \) and \( \mathcal{B} \) are defined as the union of the \( \mathcal{A}_m \) and \( \mathcal{B}_m \) sets, respectively:

\[
\mathcal{A} = \bigcup_{m \in \mathcal{P}} \mathcal{A}_m, \quad \mathcal{B} = \bigcup_{m \in \mathcal{P}} \mathcal{B}_m.
\]

Finally, we define

\[
\mathcal{C} = \bigcup_{m \in \mathcal{Q}} \mathcal{G}_m,
\]

and

\[
\mathcal{C}_m = \mathcal{G}_m \cap \mathcal{C}.
\]

**B. Optimality condition**

The objective function in (15) is convex but non-smooth as the \( \ell_{1, \infty} \) norm is non-differentiable. Therefore, problem (15) reaches its global minimum at \( w \) if and only if the sub-differential of the objective function contains the zero vector. Let \( \partial \|w\|_{1, \infty} \) denote the sub-differential of the \( \ell_{1, \infty} \) norm at \( w \). A vector \( z \in \partial \|w\|_{1, \infty} \) only if \( z \) satisfies the following conditions (details can be found in [12], [14]):

\[
\|z_{A_m}\|_1 = 1, \quad m \in \mathcal{P},
\]

(16)

\[
\sgn(z_{A_m}) = \sgn(w_{A_m}), \quad m \in \mathcal{P},
\]

(17)

\[
z_B = 0,
\]

(18)

\[
\|z_{C_m}\|_1 \leq 1, \quad m \in \mathcal{Q},
\]

(19)

where \( \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{P} \) and \( \mathcal{Q} \) are \( \beta \)-dependent sets defined on \( w \) as defined in Section III-A.

For notational convenience we drop \( \beta \) in \( R(\beta) \) and \( r(\beta) \) leaving the \( \beta \)-dependency implicit. The optimality condition is then written as

\[
Rw - r + \lambda z = 0, \quad z \in \partial \|w\|_{1, \infty}.
\]

(20)

As \( w_C = 0 \) and \( z_B = 0 \), (20) implies the three conditions

\[
R_{A \mathcal{A}} w_{\mathcal{A}} + R_{A \mathcal{B}} w_{\mathcal{B}} - r_A + \lambda z_A = 0,
\]

(21)

\[
R_{B \mathcal{A}} w_{\mathcal{A}} + R_{B \mathcal{B}} w_{\mathcal{B}} = 0,
\]

(22)

\[
R_{C \mathcal{A}} w_{\mathcal{A}} + R_{C \mathcal{B}} w_{\mathcal{B}} - r_C + \lambda z_C = 0.
\]

(23)

The vector \( w_{\mathcal{A}} \) lies in a low dimensional subspace. Indeed, by definition of \( \mathcal{A}_m \), if \( |A_m| > 1 \)

\[
|w_i| = |w_{i'}|, \quad i, i' \in \mathcal{A}_m.
\]

Therefore, for any active group \( m \in \mathcal{P} \),

\[
w_{A_m} = s_{A_m} \alpha_m
\]

(24)

where

\[
\alpha_m = \|w_{G_m}\|_1.
\]

and

\[
s_{A} = \sgn(w_{A}).
\]

Using matrix notation, we represent (24) as

\[
w_{\mathcal{A}} = S_{\mathcal{A}} \alpha.
\]

(25)
where
\[ S = \begin{pmatrix} s_{A_1} \\ \vdots \\ s_{A_{|P|}} \end{pmatrix} \]  
(26)
is a \(|A| \times |P|\) sign matrix and the vector \(a\) is comprised of \(\alpha_m, m \in P\).

The solution to (15) can be determined in closed form if the sign matrix \(S\) and sets \((A, B, C, P, Q)\) are available. Indeed, from (16) and (17)
\[ S^T z_A = 1, \]
(27)
where \(1\) is a \(|P| \times 1\) vector comprised of 1's. With (25) and (27), (21) and (22) are equivalent to
\[ S^T R_{A,A} S a + S^T R_{A,B} w_B - S^T r_A + \lambda 1 = 0, \]
\[ R_{B,A} S a + R_{B,B} w_B - r_B = 0. \]
(28)
Therefore, by defining the (a.s. invertible) matrix
\[ H = \begin{pmatrix} S^T R_{A,A} S & S^T R_{A,B} \\ R_{B,A} S & R_{B,B} \end{pmatrix}, \]
(29)
and
\[ b = \begin{pmatrix} S^T r_A \\ r_B \end{pmatrix}, v = \begin{pmatrix} a \\ w_B \end{pmatrix}, \]
(30)
(28) is equivalent to \(Hv = b - \lambda e,\) where \(e = (1^T, 0^T)^T\), so that
\[ v = H^{-1}(b - \lambda e). \]
(31)
As \(w_A = Sa\), the solution vector \(w\) can be directly obtained from \(v\) via (30). For the sub-gradient vector, it can be shown that
\[ \lambda z_{\cdot A} = r_A - (R_{A,A} S \ R_{A,B}) v, \]
(32)
\[ z_B = 0 \]
(33)
and
\[ \lambda z_{\cdot C} = r_C - (R_{C,A} S \ R_{C,B}) v. \]
(34)

C. Online update

Now we consider (15) using the results in III-B. Let \(\beta_0\) and \(\beta_1\) be two constants such that \(\beta_1 > \beta_0\). For a given value of \(\beta \in [\beta_0, \beta_1]\) define the class of sets \(S = (A, B, C, P, Q)\) and make \(\beta\) explicit by writing \(S(\beta)\). Recall that \(S(\beta)\) is specified by the solution \(f(\beta, \lambda)\) defined in (19). Assume that \(S(\beta)\) does not change for \(\beta \in [\beta_0, \beta_1]\). The following theorem propagates \(f(\beta_0, \lambda)\) to \(f(\beta_1, \lambda)\) via a simple algebraic relation.

**Theorem 1.** Let \(\beta_0\) and \(\beta_1\) be two constants such that \(\beta_1 > \beta_0\) and for any \(\beta \in [\beta_0, \beta_1]\) the solutions to (15) share the same sets \(S = (A, B, C, P, Q)\). Let \(v'\) and \(v\) be vectors defined as \(f(\beta_1, \lambda)\) and \(f(\beta_0, \lambda)\), respectively. Then
\[ v' = v + \frac{\beta_1 - \beta_0}{1 + \frac{\sigma_H}{\beta_1}} (y - \hat{y}) g, \]
(35)
and the corresponding sub-gradient vector has the explicit update
\[ \lambda z'_{\cdot A} = \lambda z_{\cdot A} + \frac{\beta_1 - \beta_0}{1 + \frac{\sigma_H}{\beta_1}} (y - \hat{y}) \{x_{\cdot A} - (R_{A,A} S \ R_{A,B}) g\}, \]
(36)
and
\[ \lambda z'_{\cdot C} = \lambda z_{\cdot C} + \frac{\beta_1 - \beta_0}{1 + \frac{\sigma_H}{\beta_1}} (y - \hat{y}) \{x_{\cdot C} - (R_{C,A} S \ R_{C,B}) g\}, \]
(37)
where \(R = R(0)\) as defined in (13), \((x, y)\) is the new sample as defined in (13) and (14), the sign matrix \(S\) is obtained from the solution at \(\beta = \beta_0\), \(H_0\) is calculated from (29) using \(S\) and \(R(0)\), and \(d, u, \hat{y}\) and \(\sigma_H^2\) are defined by
\[ d = \begin{pmatrix} S^T x_A \\ x_B \end{pmatrix}, \]
(38)
\[ g = H_0^{-1} d, \]
(39)
\[ \hat{y} = d^T v, \]
(40)
\[ \sigma_H^2 = d^T g. \]
(41)

The proof of Theorem 1 is provided in Appendix A. Theorem 1 provides the closed form update for the solution path \(f(\beta_0, \lambda) \rightarrow f(\beta_1, \lambda)\), under the assumption that the associated sets \(S(\beta)\) remain unaltered over the path.

Next, we partition the range \(\beta \in [0, 1]\) into contiguous segments over which \(S(\beta)\) is piecewise constant. Within each segment we can use Theorem 1 to propagate the solution from left endpoint to right endpoint. Below we specify an algorithm for finding the endpoints of each of these segments.

Fix an endpoint \(\beta_0\) of one of these segments. We seek a critical point \(\beta_1\) that is defined as the maximum \(\beta_1\) ensuring \(S(\beta)\) remains unchanged within \([\beta_0, \beta_1]\). By increasing \(\beta_1\) from \(\beta_0\), the sets \(S(\beta)\) will not change until at least one of the following conditions are met:

**Condition 1.** There exists \(i \in A\) such that \(z'_i = 0\);

**Condition 2.** There exists \(i \in B_m\) such that \(|u'_i| = \alpha'_m\);

**Condition 3.** There exists \(m \in P\) such that \(\alpha'_m = 0\);

**Condition 4.** There exists \(m \in Q\) such that \(|z'_m| = 1\).

Condition 1 is from (17) and (18), Condition 2 and 3 are based on definitions of \(A\) and \(P\), respectively, and Condition 4 comes from (16) and (19). Following [12], [22], the four conditions can be assumed to be mutually exclusive. The actions with respect to Conditions 1-4 are given by

**Action 1.** Move the entry \(i\) from \(A\) to \(B\):
\[ A \leftarrow A - \{i\}, B \leftarrow B \cup \{i\}; \]

**Action 2.** Move the entry \(i\) from \(B\) to \(A\):
\[ A \leftarrow A \cup \{i\}, B \leftarrow B - \{i\}; \]

**Action 3.** Remove group \(m\) from the active group list
\[ P \leftarrow P \setminus \{m\}, Q \leftarrow Q \cup \{m\}, \]
and update the related sets
\[ A \leftarrow A - A_m, C \leftarrow C \cup A_m; \]

**Action 4.** Select group \(m\)
\[ P \leftarrow P \cup \{m\}, Q \leftarrow Q \setminus \{m\}, \]
and update the related sets
\[ A \leftarrow A \cup C_m, C \leftarrow C - C_m. \]
By Theorem 1, the solution update from $\beta_0$ to $\beta_1$ is in closed form. The critical point of $\beta_1$ can be determined in a straightforward manner (details are provided in Appendix B). Let $\beta^{(k)}_1, k = 1, \ldots, 4$ be the minimum value that is greater than $\beta_0$ and meets Condition 1-4, respectively. The critical point $\beta_1$ is then

$$\beta_1 = \min_{k=1,\ldots,4} \beta^{(k)}_1.$$ 

D. Homotopy algorithm implementation

We now have all the ingredients for the homotopy update algorithm and the pseudo code is given in Algorithm 1.

Algorithm 1: Homotopy update from $f(0, \lambda)$ to $f(1, \lambda)$.

Input : $f(0, \lambda), R(0), x, y$
Output : $f(1, \lambda)$

Initialize $\beta_0 = 0$, $\beta_1 = 0$, $R = R(0)$;
Calculate $(A, B, C, P, Q)$ and $(v, \lambda z_A, \lambda z_C)$ from $f(0, \lambda)$;
while $\beta_0 < 1$ do
  Calculate the environmental variables
  $(S, H_0, d, g, \bar{y}, \sigma_B^2)$ from $f(\beta_0, \lambda)$ and $R$;
  Calculate $\{\beta^{(k)}_1\}_{k=1}^4$ that meets Condition 1-4, respectively;
  Calculate the critical point $\beta_1$ that meets Condition $k_*$: $k_* = \arg\min_k \beta^{(k)}_1$ and $\beta_1 = \beta^{(k_*)}_1$;
  if $\beta_1 \leq 1$ then
    Update $(v, \lambda z_A, \lambda z_C)$ using (35), (36) and (37);
    Update $(A, B, C, P, Q)$ by action $k_*$;
    $\beta_0 = \beta_1$;
  else
    break;
  end
$\beta_1 = 1$;
Update $(v, \lambda z_A, \lambda z_C)$ using (35);
Calculate $f(1, \lambda)$ from $v$.

Next we analyze the computational cost of Algorithm 1. The complexity to compute each critical point is summarized in Table I, where $N$ is the dimension of $H_0$. As $N = |P| + |B| \leq |A| + |B|$, $N$ is upper bounded by the number of non-zeros in the solution vector. The vector $g$ can be computed in $O(N^2)$ time using the matrix-inverse lemma [20] and the fact that, for each action, $H_0$ is at most perturbed by a rank-two matrix. This implies that the computation complexity per critical point is $O(p \max\{N, \log p\})$ and the total complexity of the online update is $O(k_2 \cdot p \max\{N, \log p\})$, where $k_2$ is the number of critical points of $\beta$ in the solution path $f(0, \lambda) \rightarrow f(1, \lambda)$. This is the computational cost required for Step 2 in Section II-C.

A similar analysis can be performed for the complexity of Step 1, which requires $O(k_1 \cdot p \max\{N, \log p\})$ where $k_1$ is the number of critical points in the solution path $f(0, \gamma \lambda) \rightarrow f(0, \lambda)$. Therefore, the overall computation complexity of the recursive $\ell_{1, \infty}$ group lasso is $O(k \cdot p \max\{N, \log p\})$, where

$k = k_1 + k_2$, i.e., the total number of critical points in the solution path $f(0, \gamma \lambda) \rightarrow f(0, \lambda) \rightarrow f(1, \lambda)$.

An instructive benchmark is to directly solve the $n$-samples problem (12) from the solution path $f(1, \infty)$ (i.e., a zero vector) $\rightarrow f(1, \lambda)$ [12], without using the previous solution $\tilde{w}_{n-1}$. This algorithm, called iCap in [12], requires $O(k' \cdot p \max\{N, \log p\})$, where $k'$ is the number of critical points in $f(1, \infty)$ $\rightarrow f(1, \lambda)$. Empirical comparisons between $k$ and $k'$, provided in the following section, indicate that iCap requires significantly more computation than our proposed Algorithm 1.

IV. Numerical simulations

In this section we demonstrate our proposed recursive $\ell_{1, \infty}$ group lasso algorithm by numerical simulation. We simulated the model $y_j = w^T_j x_j + v_j$, $j = 1, \ldots, 400$, where $v_j$ is a zero mean Gaussian noise and $w_j$ is a sparse $p = 100$ element vector containing only 14 non-zero coefficients clustered between indices 29 and 42. See Fig. 3 (a). After 200 time units, the locations of the non-zero coefficients of $w_*$ is shifted to the right, as indicated in Fig. 3 (b).

The input vectors were generated as independent identically distributed Gaussian random vectors with zero mean and identity covariance matrix, and the variance of observation noise $v_j$ is 0.01. We created the groups in the recursive $\ell_{1, \infty}$ group lasso as follows. We divide the 100 RLS filter coefficients $w$ into 20 groups with group boundaries 1, 5, 10, . . . , where each group contains 5 coefficients. The forgetting factor $\gamma$
and the regularization parameter $\lambda$ were set to 0.9 and 0.1, respectively. We repeated the simulation 100 times and the averaged mean squared errors of the RLS, sparse RLS and proposed RLS shown in Fig. 4. We implemented the standard RLS and sparse RLS using the $\ell_1$ regularization, where the forgetting factors are also set to 0.9. As [9], [10] and [15] are solving the same online optimization problem it suffices to compare the MSE of [15] to that of sparse RLS. The regularization parameter $\lambda$ was chosen to achieve the lowest steady-state error, resulting in $\lambda = 0.05$. It is important to note that, while their computational complexity is lower than that of our proposed RLS algorithm (and lower than that of [15]), the methods of [9] and [10] only approximate the exact $\ell_1$ penalized solution.

It can be seen from Fig. 4 that our proposed sparse RLS method outperforms standard RLS and sparse RLS in both convergence rate and steady-state MSE. This demonstrates the power of our group sparsity penalty. At the change point of 200 iterations, both the proposed method and sparse RLS of [15] show superior tracking performances as compared to the standard RLS. We also observe that the proposed method achieves even smaller MSE after the change point occurs. This is due to the fact that the active cluster spans across group boundaries in the initial system (Fig. 3 (a)), while the active clusters in the shifted system overlap with fewer groups.

Fig. 5 shows the average number of critical points (accounting for both trajectories in $\beta$ and $\lambda$) of the proposed algorithm, i.e., the number $k$ as defined in Section III-D. As a comparison, we implement the iCap method of [12], a homotopy-based algorithm that traces the solution path only over $\lambda$. The average number of critical points for iCap is plotted in Fig. 5, which is the number $k^i$ in Section III-D. Both the proposed algorithm and iCap yield the same solution but have different computational complexities proportional to $k$ and $k^i$, respectively. It can be seen that the proposed algorithm saves as much as 75% of the computation costs for equivalent performance.

We next investigated the effect of adding a second group of non-zero coefficients to $w_*$ and compare different algorithms for group-sparse RLS algorithms. Fig. 6(a)-(b) shows these 100 coefficients before and after a relocation of both the two groups in reverse directions at the 200th time point. The parameter settings and group allocation setting for our RLS algorithm are the same as before. The regularization parameter $\lambda$ is set to 0.03 for the proposed method. For comparison, we utilize the SpaRSA algorithm [19] implemented in a public software package\footnote{Available from http://www.lx.it.pt/~mtf/SpaRSA/} to solve group-sparse regularized RLS problems in an online manner. The SpaRSA algorithm is an efficient iterative method in which each step is obtained by solving an optimization subproblem, and it supports warm-start capabilities by choosing good initial values. In our online implementation, the current estimate is used as the initial value for solving the next estimate using the updated measurement sample. Both $\ell_1,\infty$ and $\ell_1,2$ regularizers are employed in the SpaRSA algorithm. The regularization parameter $\lambda$ of $\ell_1,\infty$-SpaRSA is set to the same value as in the proposed method and the $\lambda$ for $\ell_1,2$-SpaRSA is tuned for best steady-state MSE. As SpaRSA is an iterative algorithm, its solution is dependent on the stopping rule, which is selected based on stopping threshold on the relative change in the objective value. Two stopping threshold values, $10^{-5}$ and $10^{-3}$, are set for $\ell_1,\infty$-SpaRSA, respectively, and $10^{-5}$ is set to $\ell_1,2$-SpaRSA.

The MSE performance comparison is shown in Fig. 7 and the averaged computational time performed on an Intel Core 2.53GHz CPU is plotted in Fig. 8. It can be observed from Fig. 7 and Fig. 8 that both the MSE performance and computational time of SpaRSA algorithms depend on the stopping threshold. This is due to the fact that SpaRSA obtains approximate solutions of the regularized RLS problem rather than the exact solution attained by the proposed method. For $\ell_1,\infty$-SpaRSA, a large threshold value implies faster run time but less tracking performance in MSE as compared to a smaller stopping threshold. We observed that the MSE curve of the more computationally demanding $\ell_1,\infty$-SpaRSA will approach to that of the proposed method when the stopping threshold is very large as the two algorithms solve the same optimization problem (results not shown here). For $\ell_1,2$-SpaRSA, the algorithm achieves comparable computational time with the proposed method with the specified stopping threshold, while
Fig. 6. Coefficients for the spurious group misalignment experiment shown in Figs. 7 and 8. (a): Initial response. (b): Response after the 200th iteration. The groups for the recursive group sparse RLS algorithm (Algorithm 1) were chosen as 20 equal size contiguous groups of coefficients partitioning the range $1, \ldots, 100$.

Fig. 7. Averaged MSE of the proposed algorithm, RLS and SpaRSA algorithms [19]. The stopping thresholds for $\ell_1, \infty$-SpaRSA (a), $\ell_1, \infty$-SpaRSA (b) and $\ell_1, 2$-SpaRSA are set to $10^{-3}$, $10^{-5}$ and $10^{-5}$, respectively. The corresponding system response is shown in Fig. 6.

V. Conclusion

In this paper we proposed a $\ell_{1, \infty}$ regularized RLS algorithm for online sparse linear prediction. We developed a homotopy based method to sequentially update the solution vector as new measurements are acquired. Our proposed algorithm uses the previous estimate as a “warm-start”, from which we compute the homotopy update to the exact current solution. The proposed algorithm can process streaming measurements with time varying predictors and is computationally efficient compared to non-recursive and contemporary warm-start capable group lasso solvers. Numerical simulations demonstrated that the proposed method outperformed the standard and $\ell_1$ regularized RLS for identifying an unknown group sparse system, in terms of both tracking and steady-state mean squared error.

As indicated in Section IV, the MSE performance of the proposed method depends on the choice of group partition. The development of performance-optimal data-dependent group partitioning algorithms is a worthwhile topic for future study. Another worthwhile area of research is the extension of the proposed method to overlapping groups and other flexible partitions [23].

VI. Appendix

The authors would like to thank the reviewers for their valuable comments on the paper.

VII. Appendix

A. Proof of Theorem 1

We begin by deriving (35). According to (31),
\[ v' = H'^{-1}(b' - \lambda e'). \]
As $S$ and $(A, B, C, P, Q)$ remain constant within $[\beta_0, \beta_1]$, \[ e' = e, \]
\[ b' = b + \delta d y, \]
and
\[ H' = H + \delta d d^T, \]
where
\[ \delta = \beta_1 - \beta_0, \]
$H$ and $b$ are calculated using $S$ within $[\beta_0, \beta_1]$ and $R(\beta_0)$ and $r(\beta_0)$, respectively. We emphasize that $H$ is based on $R(\beta)$ and is different from $H_0$ defined in Theorem 1. According to the Sherman-Morrison-Woodbury formula,
\[ H'^{-1} = H^{-1} - \frac{\delta}{1 + \sigma^2 \delta} (H^{-1} d)(H^{-1} d)^T, \]
where \( \sigma^2 = d^TH^{-1}d \). Substituting (43), (44) and (45) into (42), after simplification we obtain
\[
v' = \left(H^{-1} - \frac{\delta}{1 + \sigma^2 \delta} (H^{-1}d)(H^{-1}d)^T \right) (b + \delta dy - \lambda e) = H^{-1} (b - \lambda e) + H^{-1} \delta dy - \frac{\delta^2 \delta}{1 + \sigma^2 \delta} H^{-1} \delta dy = v + \frac{\delta}{1 + \sigma^2 \delta} (y - d^Tv) H^{-1} d = v + \frac{\delta}{1 + \sigma^2 \delta} (y - \hat{y}) H^{-1} d,
\]
(46)
where \( \hat{y} = d^Tv \) as defined in (40).

Note that \( H \) is defined in terms of \( R(\beta_0) \) rather than \( R(0) \) and
\[
H = H_0 + \beta_0 d d^T,
\]
so that
\[
H^{-1} = H_0^{-1} - \frac{\beta_0}{1 + \sigma_H^2 \beta_0} g g^T,
\]
(47)
where \( g \) and \( \sigma_H^2 \) are defined by (39) and (41), respectively. As \( \sigma_H^2 = d^T g \),
\[
H^{-1} d = H_0^{-1} d - \frac{\sigma_H^2 \beta_0}{1 + \sigma_H^2 \beta_0} g.
\]
(48)
Accordingly,
\[
\sigma^2 = d^T H^{-1} d = \sigma_H^2 - \frac{\sigma_H^2 \beta_0}{1 + \sigma_H^2 \beta_0} \sigma_H^2 = \frac{\sigma_H^2}{1 + \sigma_H^2 \beta_0}.
\]
(49)
Substituting (48) and (49) to (46), we finally obtain
\[
v' = v + \frac{\delta}{1 + \sigma_H^2 \beta_1} (y - \hat{y}) g = v + \frac{\beta_1 - \beta_0}{1 + \sigma_H^2 \beta_1} (y - \hat{y}) g.
\]
Equations (36) and (37) can be established by direct substitutions of (35) into their definitions (32) and (34) and thus the proof of Theorem 1 is complete.

B. Computation of critical points

For ease of notation we work with \( \rho \), defined by
\[
\rho = \frac{\beta_1 - \beta_0}{1 + \sigma_H^2 \beta_1}.
\]
(50)
It is easy to see that over the range \( \beta_1 > \beta_0 \), \( \rho \) is monotonically increasing in \( (0, 1/\sigma_H^2) \). Therefore, (50) can be inverted by
\[
\beta_1 = \frac{\rho + \beta_0}{1 - \sigma_H^2 \rho},
\]
(51)
where \( \rho \in (0, 1/\sigma_H^2) \) to ensure \( \beta_1 > \beta_0 \).

Suppose we have obtained \( \rho(k), k = 1, ..., 4 \), \( \beta_1(k) \) can be calculated using (51) and the critical point \( \beta_1 \) is then
\[
\beta_1 = \min_{k=1,...,4} \beta_1(k).
\]

We now calculate the critical value of \( \rho \) for each condition one by one.

1) Critical point for Condition 1: Define the temporary vector
\[
t_A = (y - \hat{y}) \{ x_A - (R_{A\theta} S R_{AB}) g \}.
\]
According to (36),
\[
\lambda \lambda_A = \lambda \lambda_A + \rho t_A,
\]
Condition 1 is met for any \( \rho = \rho_1^1 \) such that
\[
\rho_1^1 = -\frac{\lambda \lambda_i}{t_i}, i \in A.
\]
Therefore, the critical value of \( \rho \) that satisfies Condition 1 is
\[
(\rho_1^1) = \min \left\{ \rho_1^1 \mid i \in A, \rho_1^1 \in (0, 1/\sigma_H^2) \right\}.
\]

2) Critical point for Condition 2: By the definition (30), \( v \) is a concatenation of \( \alpha_m \) and \( w_B, m \in P \):
\[
v^T = \left( (\alpha_m)_{m \in P}, w_{B_1}^T, ..., w_{B_{|P|}}^T \right),
\]
(52)
where \( (\alpha_m)_{m \in P} \) denotes the vector comprised of \( \alpha_m \), \( m \in P \).
Now we partition the vector \( g \) in the same manner as (52) and denote \( \tau_m \) and \( u_m \) as the counter part of \( \alpha_m \) and \( w_B \) in \( g \), i.e.,
\[
g^T = \left( (\tau_m)_{m \in P}, u_1^T, ..., u_{|P|}^T \right).
\]
Eq. (35) is then equivalent to
\[
\alpha'_m = \alpha_m + \rho \tau_m,
\]
(53)
and
\[
w'_{B,m,i} = w_{B,m,i} + \rho u_{m,i},
\]
where \( u_{m,i} \) is the \( i \)-th element of the vector \( u_m \). Condition 2 indicates that
\[
\alpha'_m = \pm w'_{B,m,i},
\]
and is satisfied if \( \rho = \rho^{(2+)}_m \) or \( \rho = \rho^{(2-)}_m \), where
\[
\rho^{(2+)}_m = \frac{\alpha_m - w_{B,m,i}}{u_{m,i} - \tau_m}, \quad \rho^{(2-)}_m = -\frac{\alpha_m + w_{B,m,i}}{u_{m,i} + \tau_m}.
\]
Therefore, the critical value of \( \rho \) for Condition 2 is
\[
\rho^{(2)} = \min \left\{ \rho^{(2+)}_m \mid m \in P, i = 1, ..., |B_m|, \rho^{(2+)}_m \in (0, 1/\sigma_H^2) \right\}.
\]

3) Critical point for Condition 3: According to (53), \( \alpha'_m = 0 \) yields \( \rho = \rho_1^{(3)} \) determined by
\[
\rho^{(3)} = -\frac{\alpha_m}{\tau_m}, m \in P,
\]
and the critical value for \( \rho^{(3)} \) is
\[
\rho^{(3)} = \min \left\{ \rho^{(3)}_m \mid m \in P, \rho^{(3)}_m \in (0, 1/\sigma_H^2) \right\}.
\]
4) Critical point for Condition 4: Define
\[ t_C = (y - \hat{y}) \{ x_C - (R_C A S \ R_C B) g \}. \]
Eq. (37) is then
\[ \lambda z_C^* = \lambda z_{m} + \rho t_{m}, \]
and Condition 4 is equivalent to
\[ \sum_{i \in C_m} | \rho t_i + \lambda z_i | = \lambda. \quad (54) \]
To solve (54) we develop a fast method that requires complexity of \( O(N \log N) \), where \( N = |C_m| \). The algorithm is given in Appendix C. For each \( m \in Q \), let \( \rho^{(4)} \) be the minimum positive solution to (54). The critical value of \( \rho \) for Condition 4 is then
\[ \rho^{(4)} = \min \left\{ \rho^{(4)}_m | m \in Q; \rho^{(4)}_m \in (0, 1/\sigma_{th}^2) \right\}. \]

C. Fast algorithm for critical condition 4

Here we develop an algorithm to solve problem (54). Consider solving the more general problem:
\[ \sum_{i=1}^{N} a_i | x - x_i | = y, \quad (55) \]
where \( a_i \) and \( x_i \) are constants and \( a_i > 0 \). Please note that the notations here have no connections to those in previous sections. Define the following function
\[ h(x) = \sum_{i=1}^{N} a_i | x - x_i |. \]
The problem is then equivalent to finding \( h^{-1}(y) \), if it exists.

An illustration of the function \( h(x) \) is shown in Fig. 9, where \( k_i \) denotes the slope of the \( i \)th segment. It can be shown that \( h(x) \) is piecewise linear and convex in \( x \). Therefore, the equation (55) generally has two solutions if they exist, denoted as \( x_{\min} \) and \( x_{\max} \). Based on piecewise linearity we propose a search algorithm to solve (55). The pseudo code is shown in Algorithm 2 and its computation complexity is dominated by the sorting operation which requires \( O(N \log N) \).

**Algorithm 2:** Solve \( x \) from \( \sum_{i=1}^{N} a_i | x - x_i | = y \).

**Input:** \( \{ a_i, x_i \}_{i=1}^{N}, y \)

**output:** \( x_{\min}, x_{\max} \)

Sort \( \{ x_i \}_{i=1}^{N} \) in the ascending order: \( x_1 \leq x_2 \leq \ldots \leq x_N \);
Re-order \( \{ a_i \}_{i=1}^{N} \) such that \( a_i \) corresponds to \( x_i \);
Set \( k_0 = -\sum_{i=1}^{N} a_i \);
for \( i = 1, \ldots, N \) do
\[ k_i = k_{i-1} + 2a_i; \]
end
Calculate \( h_1 = \sum_{i=2}^{N} a_i | x_1 - x_i |; \)
for \( i = 2, \ldots, N \) do
\[ h_i = h_{i-1} + k_{i-1} (x_i - x_{i-1}); \]
end
if \( \min k_i > y \) then
No solution;
else
if \( y > h_1 \) then
\[ x_{\min} = x_1 + (y - h_1)/k_0; \]
else
Seek \( j \) such that \( y \in [h_j, h_{j-1}]; \)
\[ x_{\min} = x_j + (y - h_j)/k_{j-1}; \]
end
else
if \( y > h_N \) then
\[ x_{\max} = x_N + (y - h_N)/k_N; \]
else
Seek \( j \) such that \( y \in [h_{j-1}, h_j]; \)
\[ x_{\max} = x_{j-1} + (y - h_{j-1})/k_{j-1}; \]
end
\]

**References**


