LASSO for Streaming Data with Adaptative Filtering

Marco Capó[1*], Aritz Pérez[1] and José A. Lozano[1,2]

[1*]Basque Center for Applied Mathematics, Bilbao, 48009, Spain.
[2]Intelligent Systems Group, Department of Computer Science and Artifitial Intelligence, University of the Basque Country UPV/EHU, Donosti, 20018, Spain.

*Corresponding author(s). E-mail(s): marco@oxcitas.com;
Contributing authors: aperez@bcamath.org; ja.lozano@ehu.es;

Abstract

Streaming data is ubiquitous in modern machine learning, and so the development of scalable algorithms to analyze this sort of information is a topic of current interest. On the other hand, the problem of $\boldsymbol{l_1}$-penalized least-square regression, commonly referred to as LASSO, is a quite popular data mining technique, which is commonly used for feature selection. In this work, we develop a homotopy-based solver for LASSO, on a streaming data context, that massively speeds up its convergence by extracting the most information out of the solution prior receiving the latest batch of data. Since these batches may show a non-stationary behavior, our solver also includes an adaptive filter that improves the predictability of our method in this scenario. Besides different theoretical properties, we additionally compare empirically our solver to the state-of-the-art: LARS, Coordinate Descent and Garrigues and Ghaoui's Data Streaming Homotopy. The obtained results show our approach to massively reduce the computational time require to convergence for the previous approaches, reducing up to 3, 4 and 5 orders of magnitude of running time with respect to LARS, Coordinate Descent and Garrigues and Ghaoui's homotopy, respectively.

Keywords: LASSO, Adaptative Filtering, Streaming Data, Homotopy

1 Introduction

Problems with high dimensionality have become common over the recent years. The high dimensionality poses significant challenges in building interpretable models with high prediction accuracy [1]. Regularized methods have established themselves as popular and effective tools through which to handle high-dimensional data [2]. Such methods employ regularization penalties as a mechanism through which to constraint the set of candidate solutions [3]. In this sense, in [4], a method for shrinkage and variable selection, called "Least Absolute Shrinkage and Selection Operator (LASSO)", is presented. It consists of a l_1-norm regularization approach that commonly leads to sparse solutions, which is a desirable property to achieve model selection, data compression and for obtaining interpretable results. For this reason, LASSO has attracted a lot of interest in the statistics, signal processing and machine learning communities [4, 5].

1.1 LASSO Optimization Problem

Formally speaking, given a matrix of predictor variables, $X \in \mathbb{R}^{n \times d}$, and a response vector, $\mathbf{y} \in \mathbb{R}^n$, LASSO consists of minimizing the following error function:

$$f_\mu^{X,\mathbf{y}}(\theta) = \frac{1}{2} \cdot \|X\theta - \mathbf{y}\|_2^2 + \mu \cdot \|\theta\|_1, \tag{1}$$

where $\mu > 0$ is a regularization parameter and $\theta \in \mathbb{R}^d$. From now on, we refer by $\theta_\mu^{X,\mathbf{y}} = \arg\min_{\theta \in \mathbb{R}^d} f_\mu^{X,\mathbf{y}}(\theta)$ to the solution of LASSO. More importantly, it should be highlighted that this optimization problem has multiple features among which we can mention:

- *Sparsity:* The solution to LASSO is typically sparse, i.e., $\theta_\mu^{X,\mathbf{y}}$ has relatively few non-zero coefficients. For this reason, LASSO is commonly used to perform feature selection [6].
- *Uniqueness:* If $\xi = \{i \in \{1, \ldots, d\} : |X_i^T(X\theta_\mu^{X,\mathbf{y}} - \mathbf{y})| = \mu\}$, then LASSO solution is unique when $\texttt{rank}(X_\xi) = |\xi|$. Otherwise, there can be multiple minimizers [7].
- *Regularization Path:* The family of solutions for LASSO, as μ varies over $(0, \infty)$, has the piecewise-linear solution path property [8]: There are values $\mu_1 > \ldots > \mu_k = 0$, such that the regularization path is a piecewise linear curve, i.e., if $\mu_{i+1} \leq \mu \leq \mu_i$, for $i \in \{1, \ldots, k-1\}$, then

$$\theta_\mu^{X,\mathbf{y}} = \frac{\mu_i - \mu}{\mu_i - \mu_{i+1}} \cdot \theta_{\mu_{i+1}}^{X,\mathbf{y}} + \frac{\mu - \mu_{i+1}}{\mu_i - \mu_{i+1}} \cdot \theta_{\mu_i}^{X,\mathbf{y}} \tag{2}$$

- *Optimality Conditions:* In general, there is a global minimum at $\theta = (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d$ for Eq.1 if and only if the sub-differential of $f_\mu^{X,\mathbf{y}}$

contains $\mathbf{0} \in \mathbb{R}^d$ [6, 7]:

$$X^T(X\theta - \mathbf{y}) + \mu \cdot \mathbf{v} = \mathbf{0}, \mathbf{v} \in \partial\|\theta\|_1, \tag{3}$$

where $\partial\|\theta\|_1$ is the sub-differential of the l_1-norm at θ: $\partial\|\theta\|_1 = \begin{cases} v_j = \texttt{sign}(\theta_j), & \text{if } \theta_j \neq 0 \\ v_j \in [-1, 1], & \text{if } \theta_j = 0 \end{cases}$.

1.2 Solving LASSO

Even though the LASSO problem can be easily stated, solving it numerically is not trivial [9]. Initially, [4] observed that LASSO can be expressed as a convex quadratic problem with linear equality constraints. This feature allows a wide variety of approaches to be used to obtain a minimizer of Eq.1. In particular, as summarized in [10], [4] used the quadratic program (QP) for least square regressions and the iteratively reweighted least square procedure with QP for generalized linear models. Afterwards, [11] proposed a faster QP algorithm for LASSO, which was implemented by [12] as `lasso2` package in `R`. Moreover, this formulation of the problem is also commonly solved using standard interior-point methods [13–16]. Unfortunately, as frequently reported in the literature, this sort of techniques can only handle small to medium-sized problems [6, 17]. A complete review on this sort of algorithms can be found in [17].

In order to cope with the computational demands of the previous approaches, other optimization techniques such as coordinate descent has been extensively used to solve LASSO [18–22]. This technique successively minimizes along coordinate directions to find the minimum of a function. At each iteration, the algorithm determines a coordinate or coordinate block via a coordinate selection rule, then minimizes over the corresponding coordinate hyperplane while fixing all other coordinates or coordinate blocks [19–21]. For the LASSO problem, an iteration of coordinate descent can be implemented in $\mathcal{O}(n \cdot d)$ time via soft-thresholding [22, 23]. On the other hand, generic methods for non-differentiable convex problems, such as the ellipsoid method or sub-gradient methods [24], can be used to solve the minimization problem, see [10, 25, 26]. Even when some of these techniques can be implemented in $\mathcal{O}(n \cdot d)$, they commonly show slow convergence which hinders their applicability in practice [25].

A last family of LASSO solvers that has gained popularity in the recent years are the homotopy methods. This technique exploits the properties of the optimization problem, in particular the regularization path property introduced in Section 1.1, to determine the LASSO solution when the regularization parameter, μ, varies [6, 8, 27–29]. It should be noted that such piecewise linearity of regularization paths are fairly common within the parametric QP formulations literature [23, 27, 30]. The most popular method in this family is LARS [8]. The algorithm begins at $\mu = \mu_{\max} = \|X^T\mathbf{y}\|_\infty$, where the solution is trivial, $\theta^{X,\mathbf{y}}_{\mu_{\max}} = \mathbf{0}$. As μ decreases, the homotopy computes a solution path θ_μ that is piecewise linear and continuous as a function of μ. Each knot in this

path corresponds to an iteration of the algorithm, in which the path's linear trajectory is altered in order to satisfy the optimality conditions [7, 8]. Analogously to solving a system via Least Squares, the computational complexity of LARS is $\mathcal{O}(d^2 \cdot \max\{n, d\})$.

1.3 Data Streaming and Adaptative Filtering

Applications involving streaming datasets are abundant, ranging from finance to cyber-security and neuroscience [3, 31, 32]. In this setting, it is assumed that batches of data, $\mathcal{X} = \{X_t\}_{t=0}^m$ and $\mathcal{Y} = \{\mathbf{y}_t\}_{t=0}^m$, arrive sequentially over time, where $X_t \in \mathbb{R}^{n_t \times d}$, $\mathbf{y}_t \in \mathbb{R}^{n_t}$. For LASSO, different works can be found in the literature for the data streaming setting. Most remarkably, [6] developed a homotopy that allows the computation of LASSO solution after observing a new data point, i.e., $m = 1$ and $n_m = 1$, for predefined regularization terms. The computational complexity of this approach for updating the LASSO solution, after observing a new data point, is $\mathcal{O}(d^2 \cdot \max\{n, d\})$, where $n = n_0 + 1$. It must be highlighted that in this work the data stream distribution is assumed invariant.

In general, it should be remarked that one major challenge that arises when dealing with these sort of applications is due to the usually common dynamic or non-stationary nature of data streams, that takes place when the underlying distribution of the data changes over time [3, 33, 34]. Fortunately, there are different ways of dealing with such a difficulty, out of which adaptive filtering is one of the most popular alternatives [35, 36]. As defined in [37], filtering is the process through which information is assimilated using data measured up to and a certain time. Adaptive filtering methods provide an elegant method through which to handle a wide range of non-stationary behavior without having to explicitly model the dynamic properties of the data stream. As shown in [3], for the LASSO error function, the adaptive filtering can be introduced by defining a set of positive weights $\mathbf{w} = \{w_t\}_{t=0}^m$ that indicates the importance of past observations, where w_t typically decays exponentially, in such a way that more relevance is provided to the latest information received. Afterwards, the weighted LASSO error function, $l_{\mu,\mathbf{w}}^{\mathcal{X},\mathcal{Y}}(\theta) = \frac{1}{2} \cdot \sum_{t=0}^m w_t \cdot \|X_t\theta - \mathbf{y}_t\|_2^2 + \mu \cdot \|\theta\|_1$, is minimized [3]. The popularity of this approach in recent years has grown, partly due to the fact that, in many scenarios, it is infeasible to model the dynamic of the data stream. The most common filtering methods discard information at a constant rate, for example determined by a fixed forgetting factor [38], which, for instance, can be achieved by setting $w_t = r^{m-t}$ for a constant value $r \in [0, 1]$. For LASSO, in the streaming data scenario with adaptive filtering, we can mainly find the work of [3, 39], which focuses their research into finding appropriate updates for the regularization parameter, μ. In particular, their proposal is to make use of stochastic gradient descent to relegate the choice of sparsity parameter to the data, in such a way that the current model optimizes the model fit on the latest batch of data, $(X_m, \mathbf{y}_m)$.

1.4 Contribution

In this work, we propose a homotopy-based solver for LASSO on a streaming data scenario, where batches of data are received sequentially over time. This new approach generalizes the algorithm presented in [6], for which the new information can be set to be a batch of points rather than a single data point. In addition, this new approach is also thought to also include an adaptive filter, which ultimately improves the predictability of our model in non-stationary settings and, hence does not require to modify the penalty term as in [3, 39]. The rest of the article is divided as follows: In Section 2, we deduce our solver and comment on some of its theoretical properties and, in Section 3, we compare our proposal empirically to the state-of-the-art.

2 LASSO for Streaming Data

In this section, we develop an efficient LASSO solver for a streaming data framework. As we previously commented, besides allowing the arrival of batches of data with multiple observations unlike the work of [6], our proposal implicitly sets an adaptive-filter on the optimization problem. Such an additional feature results advantageous especially in applications where the distribution of the data stream changes over time.

Since the batches of data $(X_t, \mathbf{y}_t)$ are received one at a time, it is convenient to find a simple way to evaluate the desired error function with adaptive filtering (Eq.2) sequentially as each batch of information arrives. In this sense, in Observation 1, we find a relation between the classical LASSO error function, Eq.1, and the weighted criterion, $l^{\mathcal{X},\mathcal{Y}}_{\mu,\mathbf{w}}(\theta)$:

Observation 1. *Given a sequence of batches,* $\mathcal{X} = \{X_t\}_{t=0}^m$, $\mathcal{Y} = \{\boldsymbol{y}_t\}_{t=0}^m$, *and a forgetting parameter* $r \in [0,1]$, *if we define the sets of batches* $(\mathcal{X}_t, \mathcal{Y}_t)$ *as follows:*

$$(\mathcal{X}_t, \mathcal{Y}_t) = \begin{cases} \left(\begin{pmatrix} X_t \\ \sqrt{r} \cdot \mathcal{X}_{t-1} \end{pmatrix}, \begin{pmatrix} \boldsymbol{y}_t \\ \sqrt{r} \cdot \mathcal{Y}_{t-1} \end{pmatrix} \right) & \text{if } t > 0 \\ (X_t, \boldsymbol{y}_t) & \text{if } t = 0 \end{cases}$$

then, for $\boldsymbol{w} = \{r^{m-t}\}_{t=0}^m$, $f_\mu^{\mathcal{X}_m,\mathcal{Y}_m}(\theta) = l^{\mathcal{X},\mathcal{Y}}_{\mu,\boldsymbol{w}}(\theta)$ *for all* $\theta \in \mathbb{R}^d$.

Observation 1 shows that we can compute the LASSO error function with adaptive filtering (with a fixed forgetting factor $r \in [0,1]$) as the classical LASSO error function just by recursively multiplying the previous set of batches by $\sqrt{r}$. Taking this into consideration, our solver seeks to determine $\theta_\mu^{\mathcal{X}_m,\mathcal{Y}_m} = \arg\min_{\theta\in\mathbb{R}^d} f_\mu^{\mathcal{X}_m,\mathcal{Y}_m}(\theta)$ efficiently via a homotopy that initially considers the solution obtained prior receiving the latest batch, $(X_m, \mathbf{y}_m)$, $\theta_\mu^{\mathcal{X}_{m-1},\mathcal{Y}_{m-1}} = \arg\min_{\theta\in\mathbb{R}^d} f_\mu^{\mathcal{X}_{m-1},\mathcal{Y}_{m-1}}(\theta)$.

It should be highlighted that penalty term, $\mu > 0$, remains invariant throughout the entire process. This is a common practice in the homotopy step for LASSO on streaming data. In any case, even when it is out of the scope of this work to determine an appropriate value for this term, the proposed solver can be further complemented with different regularization term update techniques, such as [3], by introducing a correction step prior applying the homotopy, as shown in [6], e.g., if we want to detetermine $\theta_{\mu_m}^{\mathcal{X}_m,\mathcal{Y}_m}$ via $\theta_{\mu_{m-1}}^{\mathcal{X}_{m-1},\mathcal{Y}_{m-1}}$, for $\mu_m \neq \mu_{m-1}$, we can exploit the piecewise-linearity of LASSO solution, for instance via LARS, to firstly compute $\theta_{\mu_m}^{\mathcal{X}_{m-1},\mathcal{Y}_{m-1}}$ and then we can apply our homotopy-based solver to determine $\theta_{\mu_m}^{\mathcal{X}_m,\mathcal{Y}_m}$. Taking into account the optimality conditions for the LASSO problem commented in Section 1.1, Eq.3, in the next section we develop our solver for the LASSO problem for a streaming data scenario with adaptive filtering.

2.1 Derivation of the LASSO Solver with Adaptative Filtering

Assuming that we know the optimal solution $\theta_{\mu}^{\mathcal{X}_{m-1},\mathcal{Y}_{m-1}}$ and that we are provided with a new batch of information $(X_m, \mathbf{y}_m) \in \mathbb{R}^{n_m \times d} \times \mathbb{R}^{n_m}$, we want to determine $\theta_{\mu}^{\mathcal{X}_m,\mathcal{Y}_m}$ efficiently. In order to visualize the connection between both solutions, consider the following optimization problem, for all $s \in [0,1]$:

$$\varphi(s) = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{2} \cdot \| \begin{pmatrix} s \cdot [X_m\theta - \mathbf{y}_m] \\ \sqrt{r} \cdot [\mathcal{X}_{m-1}\theta - \mathcal{Y}_{m-1}] \end{pmatrix} \|_2^2 + (r + \\ + s \cdot (1-r)) \cdot \mu \cdot \|\theta\|_1 \tag{4}$$

Observe that the function $\varphi(s) : [0,1] \to \mathbb{R}^d$ progressively gives more importance (weight) to the new batch of data, $(X_m, \mathbf{y}_m)$, as s increases. Indeed, observe that $\varphi(0) = \theta_{\mu}^{\mathcal{X}_{m-1},\mathcal{Y}_{m-1}}$ and $\varphi(1) = \theta_{\mu}^{\mathcal{X}_m,\mathcal{Y}_m}$. We will take advantage of this property to develop a homotopy that converts $\theta_{\mu}^{\mathcal{X}_{m-1},\mathcal{Y}_{m-1}}$ into $\theta_{\mu}^{\mathcal{X}_m,\mathcal{Y}_m}$ by sequentially verifying the optimality conditions of $\varphi(s)$, from $s = 0$ to $s = 1$. In particular, in this section, we verify that the function $\varphi(s)$ is in fact a piecewise smooth function of s. Afterwards, we exploit this result to derivate the proposed homotopy.

Considering the definition presented in Eq.4 and the optimality conditions of LASSO, Eq.3, we have that, for all $s \in [0,1]$, there exists $\mathbf{v}(s) \in \partial\|\varphi(s)\|_1$ for which:

$$s^2 \cdot X_m^T[X_m\varphi(s) - \mathbf{y}_m] + r \cdot \mathcal{X}_{m-1}^T[\mathcal{X}_{m-1}\varphi(s) - \mathcal{Y}_{m-1}] + \\ + (r + s \cdot (1-r)) \cdot \mu \cdot \mathbf{v}(s) = \mathbf{0} \tag{5}$$

For the sake of simplicity and without loss of generality, assume that the active set of indices/variables of $\varphi(s)$, i.e., $\mathcal{A}(s) = \{j \in \{1,\ldots,d\} : \varphi(s)_j \neq 0\}$,

appears in the first entries of $\varphi(s)$ for all $s \in [0,1]$. This is, $\varphi(s)$ is of the form $\varphi(s) = \begin{pmatrix} \widetilde{\varphi(s)} \\ \mathbf{0} \end{pmatrix}$, where $\widetilde{\varphi(s)}$ has no null entry, and $\mathbf{v}(s) = \begin{pmatrix} \widetilde{\mathbf{v}(s)} \\ \widetilde{\widetilde{\mathbf{v}(s)}} \end{pmatrix}$, where $\widetilde{\mathbf{v}(s)} = \mathtt{sign}(\widetilde{\varphi(s)})$ and $\|\widetilde{\widetilde{\mathbf{v}(s)}}\|_\infty \leq 1$ since $\mathbf{v}(s) \in \partial\|\varphi(s)\|_1$. In addition, let $\mathcal{X}_{m-1} = [\widetilde{\mathcal{X}^s_{m-1}} \ \widetilde{\widetilde{\mathcal{X}^s_{m-1}}}]$ and $X_m = [\widetilde{X^s_m} \ \widetilde{\widetilde{X^s_m}}]$ be the partition of $\mathcal{X}_{m-1}$ and X_m according to $\mathcal{A}(s)$. Taking this into consideration, the following system of equations is satisfied by $\widetilde{\varphi(s)}$ and $\widetilde{\widetilde{\mathbf{v}(s)}}$ for all $s \in [0,1]$:

$$\begin{aligned} \widetilde{\varphi(s)} = A^{-1}\mathbf{b}, \text{ where } A = s^2 \cdot \widetilde{X^s_m}^T \widetilde{X^s_m} + r \cdot \widetilde{\mathcal{X}^s_{m-1}}^T \widetilde{\mathcal{X}^s_{m-1}} \\ \text{and } \mathbf{b} = s^2 \cdot \widetilde{X^s_m}^T \mathbf{y}_m + + r \cdot \widetilde{\mathcal{X}^s_{m-1}}^T \mathcal{Y}_{m-1} - (r + s \cdot (1-r)) \cdot \mu \cdot \widetilde{\mathbf{v}(s)} \end{aligned} \tag{6}$$

$$\begin{aligned} \widetilde{\widetilde{\mathbf{v}(s)}} = -((r + s \cdot (1-r)) \cdot \mu)^{-1} \cdot [s^2 \cdot \widetilde{\widetilde{X^s_m}}^T [\widetilde{X^s_m}\widetilde{\varphi(s)} - \mathbf{y}_m] + \\ + r \cdot \widetilde{\widetilde{\mathcal{X}^s_{m-1}}}^T [\widetilde{\mathcal{X}^s_{m-1}}\widetilde{\varphi(s)} - \mathcal{Y}_{m-1}]] \end{aligned} \tag{7}$$

In words, if we know the active set of indices, $\mathcal{A}(s)$, and signs of the coefficients, $\widetilde{\mathbf{v}(s)}$, we can compute the solution to LASSO, $\varphi(s)$, in closed form. Furthermore, it should be highlighted that $\varphi(s)$ is also a piecewise smooth function of $s \in [0,1]$, see Theorem 1 and Corollary 1.

Theorem 1. *If* $\|\widetilde{\widetilde{\boldsymbol{v}(s^*)}}\|_\infty < 1$ *is satisfied for a given* $s^* \in [0,1]$*, then there is* $\varepsilon > 0$*, such that* $\widetilde{\boldsymbol{v}(s)} = \widetilde{\boldsymbol{v}(s^*)}$ *and* $\|\widetilde{\widetilde{\boldsymbol{v}(s)}}\|_\infty < 1$*, for all* $s \in [s^*, s^* + \varepsilon)$*.*

Theorem 1 shows that, given a solution $\varphi(s^*)$, for a certain $s^* \in [0,1)$, there is an interval, $s \in [s^*, s^* + \varepsilon)$, in which the set of active variables, $\mathcal{A}(s)$, and the signs of the coefficients, $\widetilde{\mathbf{v}(s)}$, remain invariant, and so, $\varphi(s)$ can be computed in closed form using Eq.6, for $s \in [s^*, s^* + \varepsilon)$. In this sense, it is of our interest to determine those points in $[0,1]$ for which the active set of indices changes, which are commonly referred to in the literature as transition points [6, 8]: In particular, a homotopy-based solver commonly intends to efficiently determine such transition points and, more importantly, the corresponding modification on the active set of indices, until the desired solution, in our case $\mathcal{A}(1)$ and $\varphi(1) = \theta_\mu^{\mathcal{X}_m, \mathcal{Y}_m}$, can be computed. In the following section, we extend on the computation of these points for our setting.

The rest of the section is divided in three parts. First, in Section 2.1.1, we will comment on the challenges of effectively computing transition points for the problem proposed in Eq. 4 and discuss some scenarios in which such a computation is tractable. In Section 2.1.2, we will deduce a heuristic, Algorithm 1, that we will use to compute the changes in the active set of indices regardless

if the transition point can be computed explicitly. Finally, in Section 2.1.3, we will formaly define our homotopy-based approach LASSO solver with adaptive filtering (Algorithm 2).

2.1.1 Challenges in Computing the Transition Points

As we commented in the previous section, the proposed homotopy-based solver for the LASSO with adaptive filtering problem, Eq.4, sequentially determines those intervals within $[0, 1]$ for which active set of indices and corresponding signs remain invariant. In particular, given a solution $\varphi(s^*)$, for some $s^* \in [0, 1)$, we are primarily interested in determining the closest transition point $s > s^*$. The goal of this section is to comment on the challenges of computing such transition points explicitly. A first step in this direction is given in the following result:

Corollary 1. *Given a solution* $\widetilde{\varphi(s^*)} = A^{-1}\boldsymbol{b}$, *for* $s^* \in [0, 1)$, *where* A *and* $\boldsymbol{b}$ *are defined by Eq.6, and* $\Lambda = \widetilde{X_m^{s^*}}^T \widetilde{X_m^{s^*}} A^{-1}$, *then*

$$\widetilde{\varphi(s)} = \widetilde{\varphi(s^*)} - (s - s^*) \cdot A^{-1}[\mathbb{I} + (s^2 - s^{*2}) \cdot \Lambda]^{-1}[(s + s^*) \cdot \widetilde{X_m^{s^*}}^T \widetilde{e} + \\ + (1 - r) \cdot \mu \cdot \widetilde{\boldsymbol{v}(s^*)}], \quad (8)$$

where $\widetilde{e} = \widetilde{X_m^{s^*}} \widetilde{\varphi(s^*)} - \boldsymbol{y}_m$, *for all* $s \in [s^*, s^* + \varepsilon)$, *and* ε *defined in Theorem 1.*

Making use of Corollary 1, given s^* and $\mathcal{A}(s^*)$, we could implicitly determine the next transition point s, for which some entry of $\widetilde{\varphi(s)}$ is zero. Such an entry would ultimately exit the current set of active indices. Ideally, we should be able to determine such value s explicitly, however, in general, computing explicitly the inverse matrix $[\mathbb{I} + (s^2 - s^{*2}) \cdot \Lambda]^{-1}$, as a function of s, is a difficult task. In spite of this, in the following results we comment on different scenarios under which this task is attainable:

Observation 2. *If the eigenvalues of the matrix* $(s^2 - s^{*2}) \cdot \Lambda$ *satisfy* $\max \lambda_{(s^2 - s^{*2}) \cdot \Lambda} < 1$, *then we can determine, for each variable* $j \in \mathcal{A}(s^*)$, *the value* s *for which* $\widetilde{\varphi(s)}_j = 0$, *by solving*

$$\sum_{i=0}^{\infty} (s^{*2} - s^2)^{i+1} \cdot \Gamma_{1j}^i - (s^{*2} - s^2)^i \cdot (s^* - s) \cdot \Gamma_{2j}^i = -\widetilde{\varphi(s^*)}_j, \quad (9)$$

where $\Gamma_1^i = A^{-1}\Lambda^i \widetilde{X_m^{s^*}}^T \widetilde{e}$ *and* $\Gamma_2^i = (r - 1) \cdot \mu \cdot A^{-1}\Lambda^i \widetilde{X_m^{s^*}}^T \widetilde{\boldsymbol{v}(s^*)}$.

Observation 2 shows that, under some mild-conditions over the eigenvalues of $(s^2 - s^{2*}) \cdot \Lambda$, we can determine the transition points by finding the zeros of the infinite series presented in Eq.9. Moreover, observe that we can truncate such a inifinite sum up to a low-degree polynomial and still get an accurate

guess of the transition point, s, since $s, s^* \in [0,1]$ and so $(s^2 - s^{*2})^i$ descends rapidly towards zero, for $i > 1$. Furthermore, in the scenario $n_m = 1$, i.e., when only one observation is received at the time as in [6], one can easily compute the transition points explicitly from the data:

Theorem 2. *Given a solution* $\widetilde{\varphi(s^*)} = A^{-1}\boldsymbol{b}$*, for* $s^* \in [0,1)$*, where* A *and* $\boldsymbol{b}$ *are defined by Eq.6. If* $n_m = 1$*,* $\boldsymbol{u} = A^{-1}\widetilde{X_m^{s^*}}^T$*,* $\alpha = \widetilde{X_m^{s^*}}\boldsymbol{u}$ *and* $\boldsymbol{z} = (r-1)\cdot\mu\cdot A^{-1}\widetilde{\boldsymbol{v}(s^*)}$*, then* $\widetilde{\varphi(s)}_j = 0$*, for*

$$s = \frac{\boldsymbol{z}_j \cdot \boldsymbol{m}_j}{2} \pm \left(s^{*2} - \varphi(s^*)_j \cdot \boldsymbol{m}_j + \boldsymbol{z}_j \cdot \boldsymbol{m}_j \cdot \left(\frac{\boldsymbol{m}_j}{4} + s^*\right)\right)^{\frac{1}{2}},$$

where $\boldsymbol{m}_j = (\alpha \cdot \varphi(s^*)_j - \widetilde{e} \cdot \boldsymbol{u}_j)^{-1}$*, for all* $j \in \mathcal{A}(s^*)$*.*

Observe that the result presented in Theorem 2 is a generalization of the result shown in [6], in terms of the computation of transition points: for the latter, no adaptive filtering is considered, i.e., $\mathbf{z} = \mathbf{0}$.

Unfortunately, as commented throughout this section, computing such transition points explicitly from the data still remains an untractable problem in most scenarios. In spite of this, we should recall that, given the optimality conditions presented in Eq.6-7, we only require to know the adequate set of active indices, $\mathcal{A}(1)$, and their corresponding signs, $\widetilde{\mathbf{v}(1)}$, to compute the desired solution, $\theta_\mu^{\mathcal{X}_m, \mathcal{Y}_m}$. Since $\mathcal{A}(s)$ changes throughout $s \in [0,1]$, we can instead focus our analysis on determining such modifications rather than finding out explicitly when they occur (transition point). In the next section, we comment on an efficient approach to determine the appropriate modification of the active set of indices for $s \in [0,1]$.

2.1.2 About the Active Set of Indices

Given a solution $\varphi(s^*)$, for some $s^* \in [0,1)$, in this section we develop a method to determine the earliest modification of the current set of active indices, i.e., $\mathcal{A}(s) \neq \mathcal{A}(s^*)$, for $s > s^*$, which is formaly presented in Algorithm 1. For the sake of simplicity, we assume $|\mathcal{A}(s)| = |\mathcal{A}(s^*)| \pm 1$, meaning that at most one variable violates the optimality conditions at $s > s^*$, given the set of active indices, $\mathcal{A}(s^*)$, and associated array of signs, $\widetilde{\mathbf{v}(s^*)}$, which is the event that occurs almost surely. The extension to the other case is straightforward.

To start and in order to illustrate the evolution of the active set of indices, in Fig.1, we plot the optimality conditions (optimal value of the regression coefficient, $\varphi(s)$ and sub-differential, $\mathbf{v}(s)$), in $s \in [0,1]$, for a toy example with $m = 1$, $n_0 = 5000$, $n_1 = 2000$, $d = 3$, $r = 0.50$ and $\mu = 0.25 \cdot \mu_{\max}$. In particular, the optimal value of the coefficients, $\varphi(s)_j$, associated to the active variables, $j \in \mathcal{A}(s)$, are plotted in blue, while, for the non-active, we show the sub-differential values, $\mathbf{v}(s)_j$, in red. Furthermore, in dashed black lines, we observe the critical values for both terms: i) $\varphi(s)_j = 0$, which implies the

modification of j to the non-active variables, and ii) $|\mathbf{v}(s)_j| > 1$, in which case $j \in \mathcal{A}(s)$.

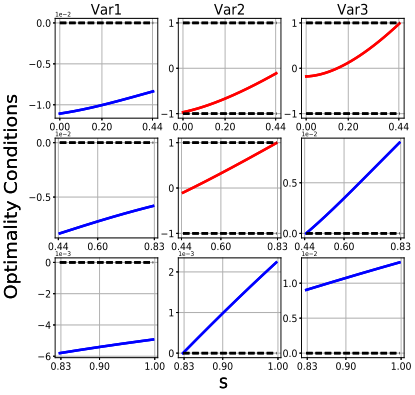

Figure 1: Evolution of $\widetilde{\varphi(s)}$ (blue curves) and $\widetilde{\widetilde{\mathbf{v}(s)}}$ (red curves), for $s \in [0, 1]$, for `Var1`, `Var2` and `Var3`.

According to Fig.1, we observe that at the beginning, $s = 0$, just `Var1` is active. However, at $s = 0.44$, `Var3` joins the active set of variables and, finally, at $s = 0.83$, `Var2` becomes active as well. Therefore, the active variables of the corresponding LASSO with adaptative filtering problem are `Var1`, `Var2` and `Var3`, which differs from the original set of active indices. The goal of this section is then to develop a low computational-cost complexity algorithm to determine such modifications over the set of variables.

To start our analysis, we first define the functions $\zeta_{s^*}(s)$ and $\Psi_{s^*}(s)$, which will allow us to verify if the optimality conditions, Eq.6-7, for a point, $s \in (s^*, 1]$, are held for the optimal setting obtained at s^*. As we can see in line 5 of Algorithm 1, verifying such conditions is the initial step of our active set update heuristic:

$$\zeta_{s^*}(s) = [s^2 \cdot C + A]^{-1}[\mathbf{b} + s^2 \cdot \mathbf{d} - s \cdot (1 - r) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)}] \tag{10}$$

$$\Psi_{s^*}(s) = -((r + s \cdot (1 - r)) \cdot \mu)^{-1} \cdot [s^2 \cdot [D\zeta_{s^*}(s) - \mathbf{e}] + B\zeta_{s^*}(s) - \mathbf{c}], \tag{11}$$

where $A = r \cdot \widetilde{\mathcal{X}^{s^*}_{m-1}}^T \widetilde{\mathcal{X}^{s^*}_{m-1}}$, $B = r \cdot \widetilde{\widetilde{\mathcal{X}^{s^*}_{m-1}}}^T \widetilde{\mathcal{X}^{s^*}_{m-1}}$, $C = \widetilde{X^{s^*}_m}^T \widetilde{X^{s^*}_m}$, $D = \widetilde{\widetilde{X^{s^*}_m}}^T \widetilde{X^{s^*}_m}$ and $\mathbf{b} = r \cdot \widetilde{\mathcal{X}^{s^*}_{m-1}}^T \mathcal{Y}_{m-1} - r \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)}$, $\mathbf{c} = r \cdot \widetilde{\widetilde{\mathcal{X}^{s^*}_{m-1}}}^T \mathcal{Y}_{m-1}$, $\mathbf{d} = \widetilde{X^{s^*}_m}^T \mathbf{y}_m$, $\mathbf{e} = \widetilde{\widetilde{X^{s^*}_m}}^T \mathbf{y}_m$. From now on, we refer to the collection of these arrays as $\mathcal{M} = \{A, B, C, D, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}\}$.

It must be remarked that, if the optimality conditions are satisfied at $s > s^*$, i.e., $\texttt{sign}(\zeta_{s^*}(s)) = \widetilde{\mathbf{v}(s^*)}$ and $\|\Psi_{s^*}(s)\|_\infty \leq 1$, then $\widetilde{\varphi(s)} = \zeta_{s^*}(s)$ and $\widetilde{\widetilde{\mathbf{v}(s)}} = \Psi_{s^*}(s)$. In order to formalize the verification of the optimality conditions, we define the following sets of indices:

$$\mathcal{A}^s_{s^*} = \{j \in \mathcal{A}(s^*) : \texttt{sign}(\zeta_{s^*}(s)_j) \neq \widetilde{\mathbf{v}(s^*)}_j\} \tag{12}$$

$$\mathcal{I}^s_{s^*} = \{j \in \{1, \ldots, d\} \setminus \mathcal{A}(s^*) : |\Psi_{s^*}(s)_j| > 1\} \tag{13}$$

Observe that $\mathcal{A}^s_{s^*}$ is composed by those indices in the active set for which $\zeta_{s^*}(s)$ does not have the expected sign, while $\mathcal{I}^s_{s^*}$ contains the inactive indices that do not satisfy the sub-differential. Furthermore, we define $\mathcal{V}^s_{s^*} = \mathcal{A}^s_{s^*} \cup \mathcal{I}^s_{s^*}$ as the set of indices violating either optimality condition. For this reason, the optimality conditions are satisfied at $s > s^*$, for $\mathcal{A}(s^*)$, if and only if $|\mathcal{V}^s_{s^*}| = 0$. Taking all this into account, given any $\varphi(s^*)$, in Algorithm 1, we initially compute $\zeta_{s^*}(1)$ and $\Psi_{s^*}(1)$ to verify if $|\mathcal{V}^s_{s^*}| = 0$. If this is the case, then $\theta^{\mathcal{X}_m, \mathcal{Y}_m}_{\mu} = \begin{pmatrix} \zeta_{s^*}(1) \\ \mathbf{0} \end{pmatrix}$ and the algorithm stops. Otherwise, we follow a bisection-like search, in order to determine a point $s \in (s^*, 1)$ for which $|\mathcal{V}^s_{s^*}| \leq 1$:

- `Case 1:` $|\mathcal{V}^s_{s^*}| = 0$ (line 6 in Algorithm 1). In this case $\mathcal{A}(s) = \mathcal{A}(s^*)$ and therefore, if s^{**} is the previous value of s, i.e., the last explored value for which $\mathcal{A}(s^{**}) \neq \mathcal{A}(s^*)$, we can update $s^* = s$ and re-start the search with $s = \frac{1}{2} \cdot (s^* + s^{**})$ until `Case 2` occurs.
- `Case 2:` $|\mathcal{V}^s_{s^*}| = 1$ (line 13 in Algorithm 1). In this scenario either: i) $|\mathcal{A}^s_{s^*}| = 1$ or ii) $|\mathcal{I}^s_{s^*}| = 1$. The variable not satisfying the optimality condition at i), must exit the active set of indices, while it must included in the active set in ii). Hence, we must verify if $\mathcal{A}(s) = \mathcal{A}(s^*) \cup \mathcal{I}^s_{s^*} \setminus \mathcal{A}^s_{s^*}$ holds. If so, we take $s^* = s$ and $\mathcal{A}(s^*) = \mathcal{A}(s)$ and re-start the search with $s = 1$. If not, we take $s = \frac{1}{2} \cdot (s + s^*)$.

For the example presented in Fig.1, our method initially evaluates the optimality of $\zeta_0(1)$ and $\Psi_0(1)$. Since $|\mathcal{V}^1_0| = 2$, it considers $s = 0.50$, for which $|\mathcal{V}^{0.50}_0| = 1$, i.e., `Case 2`. Since $\mathcal{I}^{0.50}_0 = \{\texttt{Var3}\}$, we verify that $\mathcal{A}(0.50) = \mathcal{A}(0) \cup \mathcal{I}^{0.50}_0 = \{\texttt{Var1}, \texttt{Var3}\}$. Analogously, we then verify that $|\mathcal{V}^1_{0.50}| = |\mathcal{I}^1_{0.50}| = 1$, where $\mathcal{I}^1_{0.50} = \{\texttt{Var2}\}$, and so, we conclude that $\mathcal{A}(1) = \mathcal{A}(0.50) \cup \mathcal{I}^1_{0.50} = \{\texttt{Var1}, \texttt{Var2}, \texttt{Var3}\}$. In Algorithm 1, we present the proposed method for updating the active set of indices based on the previously described methodology.

The computational load of Algorithm 1 is largely dominated by the computation of $\zeta_{s^*}(s)$ and $\Psi_{s^*}(s)$. Since the arrays contained in $\mathcal{M}$ are already pre-computed to determine the input $\varphi(s^*)$, evaluating $\zeta_{s^*}(s)$ can be done in $\mathcal{O}(d^3_{s^*})$ time, where $d_{s^*} = |\mathcal{A}(s^*)|$. Furthermore, $\Psi_{s^*}(s)$ is computed in $\mathcal{O}(d_{s^*} \cdot (d - d_{s^*}))$ and so, the overall complexity of Algorithm 1 is $\mathcal{O}(d_{s^*} \cdot \max\{d - d_{s^*}, d^2_{s^*}\})$.

Algorithm 1 `Active Set Updater`

Input: Solution $\varphi(s^*)$, for $s^* \in [0, 1)$, and associated collection of arrays, $\mathcal{M}$.
Output: Solution $\varphi(s)$, for some $s \in (s^*, 1]$.
Set $s = 1$.
while `not Transition` **do**
    Compute $\zeta_{s^*}(s)$, $\Psi_{s^*}(s)$ and $\mathcal{V}^s_{s^*}$ (Eq.10-11).
    if $|\mathcal{V}^s_{s^*}| == 0$ **then**
        Set $s^* = s$.
        if $s == 1$ **then**
            `Transition`
        else
            Set $s = \frac{1}{2} \cdot (s^* + s^{**})$.
        end if
    else if $|\mathcal{V}^s_{s^*}| == 1$ **then**
        if $\mathcal{A}(s) == \mathcal{A}(s^*) \cup \mathcal{I}^s_{s^*} \setminus \mathcal{A}^s_{s^*}$ **then**
            Set $s^* = s$ and $\mathcal{A}(s^*) = \mathcal{A}(s)$. `Transition`.
        else
            Set $s^{**} = s$ and $s = \frac{1}{2} \cdot (s^* + s)$.
        end if
    else
        Set $s^{**} = s$ and $s = \frac{1}{2} \cdot (s^* + s)$.
    end if
end while
Return $s^* \leq 1$, active set of indices, $\mathcal{A}(s^*)$, and solution, $\varphi(s^*) = \begin{pmatrix} \zeta_{s^*}(s^*) \\ \mathbf{0} \end{pmatrix}$.

It should be noted that once the set of active indices is updated via Algorithm 1, one must also update the arrays in $\mathcal{M}$. Fortunately, after applying Algorithm 1, only one variable enters or exits the active set of indices. For this reason, most of the arrays update can be achieved by removing/concatenating rows and columns of the previous arrays. In particular, in this step we have two cases:

- The new set of active indices is given by $\mathcal{A}(s^*) \setminus \mathcal{A}^s_{s^*}$, i.e., one active variable $j \in \mathcal{A}(s^*)$ joins the inactive set. This is the simplest scenario, since both A and C are updated by eliminating their j^{th} row and column. Such a j^{th} row of A and C (except its j^{th} entry) are then concatenated as the last row of arrays B and D, respectively, to update them. Analogously, $\mathbf{b}$ and $\mathbf{d}$ are updated by eliminating their j^{th} entry, which is then concatenated as the last entry of $\mathbf{c}$ and $\mathbf{e}$. Therefore, the update of $\mathcal{M}$ can be done in $\mathcal{O}(1)$ time in this case.
- The new set of active indices is given by $\mathcal{A}(s^*) \cup \mathcal{I}^s_{s^*}$, i.e., one inactive variable $j \notin \mathcal{A}(s^*)$ joins the active set. This scenario is slightly more difficult

as we need to compute the dot product between the j^{th} column of $\widetilde{\widetilde{X_m^{s^*}}}$ (or $\widetilde{\widetilde{\mathcal{X}_{m-1}^{s^*}}}$) and all the columns of $\widetilde{X_m^{s^*}}$ (or $\widetilde{\mathcal{X}_{m-1}^{s^*}}$) and itself and add them as the last row and column of A and C, respectively, which has a $\mathcal{O}(n \cdot d_{s^*})$ cost, where $n = \sum_{t=0}^{m} n_t$. Analogously, for B and D, we just need to compute the dot product of between the j^{th} column of $\widetilde{\widetilde{X_m^{s^*}}}$ (or $\widetilde{\widetilde{\mathcal{X}_{m-1}^{s^*}}}$) and all the other columns of $\widetilde{\widetilde{X_m^{s^*}}}$ (or $\widetilde{\widetilde{\mathcal{X}_{m-1}^{s^*}}}$), which is $\mathcal{O}(n \cdot (d - d_{s^*}))$. Following the same reasoning, both $\mathbf{b}$ and $\mathbf{d}$ are updated in $\mathcal{O}(n)$ time, and $\mathbf{c}$ and $\mathbf{e}$ in $\mathcal{O}(1)$ time. Therefore, the update of $\mathcal{M}$ can be done in $\mathcal{O}(n \cdot d)$ time in this case.

Taking all these steps into account, in the following section we formalize the proposed solver for LASSO with adaptive filtering for streaming data.

2.1.3 Adaptative Filtering LASSO Homotopy

In this section, we present the implementation of the Adaptative Filtering LASSO Homotopy (AFLH) in Algorithm 2. As commented earlier, the goal of this approach is to make the corresponding transformations over the active set of variables of $\varphi(0) = \theta_\mu^{\mathcal{X}_{m-1}, \mathcal{Y}_{m-1}}$ to determine $\varphi(1) = \theta_\mu^{\mathcal{X}_m, \mathcal{Y}_m}$ by exploting the piecewise smoothness property of $\varphi(s)$.

Using the bisection-based approach presented in Algorithm 1, and given a solution $\varphi(s^*)$, for $s^* \in [0, 1)$, we evaluate the optimality conditions, Eq.6-7, for $s \in (s^*, 1]$, with the set of active variables $\mathcal{A}(s^*)$, Eq.10-11, in such a way that we can determine the appropriate modification of the active set of indices. The process ends when we can determine $\mathcal{A}(1)$, in which case we can compute $\varphi(1) = \theta_\mu^{\mathcal{X}_m, \mathcal{Y}_m}$.

Algorithm 2 `AFLH Algorithm`

1: **Input:** Solution $\varphi(0) = \theta_\mu^{\mathcal{X}_{m-1}, \mathcal{Y}_{m-1}}$ and associated collection of arrays $\mathcal{M} = \{A, B, \mathbf{b}, \mathbf{c}\}$.
2: **Output:** Solution $\varphi(1) = \theta_\mu^{\mathcal{X}_m, \mathcal{Y}_m}$.
3: Set $s^* = 0$ and compute $C, D, \mathbf{d}, \mathbf{e}$ and add them to $\mathcal{M}$.
4: **while** $s^* < 1$ **do**
5: Compute $\zeta_{s^*}(s)$, $\Psi_{s^*}(s)$ and $\mathcal{V}_{s^*}^s$ (Eq.10-11).
6: Update $(s^*, \mathcal{A}(s^*), \varphi(s^*))$ by appying Algorithm 1 on $\varphi(s^*)$ and $\mathcal{M}$.
7: Update the collection of arrays $\mathcal{M}$ using $\mathcal{A}(s^*)$.
8: **end while**
9: **Return** $\theta_\mu^{\mathcal{X}_m, \mathcal{Y}_m} = \varphi(1)$ and active set $\mathcal{A}(1)$.

As we previously commented, the homotopy proposed in Algorithm 2 alternates between finding the transformation of the active set of indices via Algorithm 1, which is $\mathcal{O}(d^* \cdot \max\{d - d^*, d^{*2}\})$, where $d^* = \max_{s \in [0,1]} |\mathcal{A}(s)|$, and its corresponding effect on the collection of arrays, $\mathcal{M}$, discussed at the end

of Section 2.1.2, which in the worst case scenario is $\mathcal{O}(n \cdot d)$. More importantly, we must point out that the only arrays that are known at the beginning of Algorithm 2 are those related to $\mathcal{X}_{m-1}$ and $\mathcal{Y}_{m-1}$, for $s^* = 0$, i.e., A, B, $\mathbf{b}$ and $\mathbf{c}$. For this reason, we need to additionally compute C, D, $\mathbf{d}$ and $\mathbf{e}$ to construct $\mathcal{M}$. This can be done in $\mathcal{O}(n_m \cdot d^* \cdot \max\{d^*, d - d^*\})$ time. Hence, Algorithm 2 is, in the worst case scenario, $\mathcal{O}(\max\{n \cdot d, d^* \cdot \max\{d^{*2}, n_m \cdot \max\{d^*, d - d^*\}\}\})$. This analysis can be further simplified if we assume a fairly common scenario: $d^* < d - d^*$ and $d^* < n_m$, in which case we have a $\mathcal{O}(\max\{n \cdot d, n_m \cdot d^* \cdot (d - d^*)\})$ complexity. Observe that such a complexity is much lower than the one obtained by most of the solvers presented in Sections 1.2-1.3, which tend to be $\mathcal{O}(d^2 \cdot \max\{n, d\})$.

3 Experiments

In a similar manner to the experimental setting proposed in [6], in this section we empirically compare our proposal, the Adaptative Filtering LASSO Homotopy algorithm (**AFLH**), to Coordinate Descent, taking the corresponding LASSO solution prior receiving the last batch as warm-start (**CD_ws**) and with random initialization (**CD**), LARS method (**LARS**) and the One-Observation-At-A-Time LASSO Homotopy algorithm proposed by Garrigues & Ghaoui in [6] (**OOLH**)[1].

In order to observe the behavior of the considered algorithms for the different parameters of the problem, we first analyze the performance of **AFLH**, **CD**, **CD_ws**, **LARS** and **OOLH** for different regresion problems generated via the `make_regression` routine of the `Scikit-Learn` library, for $n \in \{500, 5000, 50000\}$ and $d \in \{100, 1000, 10000\}$[2]. In particular, for our experimental setting, we consider the case $m = 1$ and so, given a data set, $(\mathcal{X}, \mathcal{Y})$, we split it into two data sets: we first select $100 \cdot (1 - n_0)\%$ of the instances uniformly at random as the set of previous batches, $(\mathcal{X}_{m-1}, \mathcal{Y}_{m-1})$, and the the rest of the instances define the new batch of information, $(X_m, \mathbf{y}_m)$, with with $n_0 \in \{0.01, 0.05, 0.10\}$. Additionally, we set $r \in \{0.50, 0.75, 1\}$ and $\mu \in \{0.001 \cdot \mu_{\max}, 0.005 \cdot \mu_{\max}, 0.01 \cdot \mu_{\max}, 0.05 \cdot \mu_{\max}, 0.10 \cdot \mu_{\max}\}$. Each experiment is repeated five times.

In Figs 2-6, we show the relative computational times for all the methods (proportion of running time of a given method with respect to the method that converges the fastest) for all the considered parameters of the problem.

[1] In order to make a fair comparison in terms of running times, all considered algorithms are implemented in Python: For **OOLH**, we are using the https://github.com/pierreg/reclasso published in [6]. On the other hand, **AFLH** was also coded in https://github.com/MarcoVCapo/LASSO/blob/main/AFLH.py and, finally, for **CD** and **LARS**, we used the `Scikit-Learn` implementations of them.

[2] In the Supplementary Material, we additionally show results for real data sets obtained from the `UCI repository`.

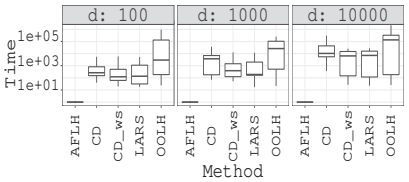

Figure 2: Relative computational time w.r.t. the dimensionality of the problem, d.

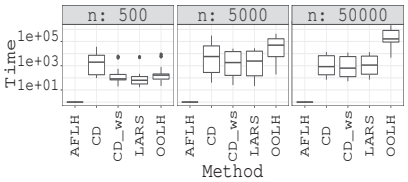

Figure 3: Relative computational time w.r.t. the number of instances, n.

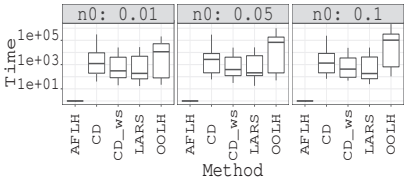

Figure 4: Relative computational time w.r.t. batch size, n_0.

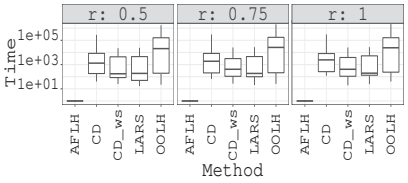

Figure 5: Relative computational time w.r.t. the forgetting term, r.

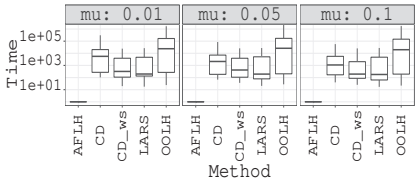

Figure 6: Relative computational time w.r.t. the penalty term, μ.

The results presented in the previous plots show that, regardless of scenario presented, **AFLH** not only converged to the optimal LASSO solution in the lowest running time, but it also massively reduces the computational running times of **CD**, **CD_ws**, **LARS** and **OOLH**. For instance, in the case of the largest dimensionality, $d = 10000$, **AFLH** is, on average, a staggering 4.0×10^5, 7.9×10^3, 8.6×10^3 and 3.1×10^5 times faster than **CD**, **CD_ws**, **LARS** and **OOLH**, respectively.

In general, we observe that the factor for which the relative running time of most algorithms seems mostly affected is the dimensionality of the problem, d, see Fig.2. In particular, we have that the ratio between the average relative running times of **CD**, **CD_ws**, **LARS** and **OOLH**, for $d = 10000$ over $d = 100$, is 91.2, 28.3, 19.6 and 2.15 for **CD**, **CD_ws**, **LARS** and **OOLH**, respectively. For all the other considered factors, the relative computational times commonly seems to keep a similar behavior with respect to that obtained by **AFLH**, see Figs.3-6.

In comparison to the other streaming homotopy, **OOLH**, we observe that as the batch size is larger, n_0, its relative running times with respect to **AFLH** increases signifantly, see Table 1. This is expected since **OOLH** re-computes the homotopy one instance at a time, which makes it especially uncompetitive when the batch size increases.

Table 1: Average relative running times of **OOLH** w.r.t. **AFLH**.

Method	$n_0 = 0.01$	$n_0 = 0.05$	$n_0 = 0.10$
OOLH/AFLH	4.0×10^4	1.8×10^5	3.2×10^5

One last factor that allows us to see the benefits of using the current solution as warm-start of a LASSO solver when a new batch of data arrives can be seen in Table 2.

Table 2: Average relative number of iterations of **CD** w.r.t. **CD_ws**.

Method	$d = 100$	$d = 1000$	$d = 10000$
CD/CD_ws	2.1×10^0	1.6×10^1	1.3×10^2

In Table 2, we observe the speed-up achieved -in terms of the number of iterations of **CD**- by taking the previous solution as initialization of the algorithm. For $d = 10000$, **CD** converged over 100 times faster just by making such a modification.

Conclusions

In this work, we present a fast LASSO solver for data streams that can also consider adaptive filters, called AFLH. Our proposal is a homotopy-based algorithm that makes use of the LASSO solution obtained prior receiving the last batch of information to speed up the convergence of our method over the entire data set. Not only is AFLH an extension of the well-known homotopy-based solver proposed by Garrigues & Ghaoui in [6] to a more general scenario in which the explicit computation of transition points is not straightforward, but it massively reduces the running times of this and other fairly popular methods such as Coordinate Descent and LARS -over 10^5 times faster on average for data sets with more than $d = 10000$ features-. Furthermore, throughout the article, besides presenting different theoretical guarantees of AFLH, we make a detailed analysis of the computational complexity of our proposal and optimize the implementation of each of its steps.

Appendix

This appendix is divided in two sections. First, in Section A, we show the proofs of the all the theoretical results presented in the main article and comment on their importance. Afterwards, in Section B, we add complementary empirical results on artificial and real data sets.

A Proofs

In this section, we sketch the proofs of the theorems presented throughout the article. The first result, Observation 1, allows us to relate the weighted (adaptive filter) LASSO-based error function, Eq.2, to the common LASSO error function, Eq.1. We will use this result to compare our proposal to the state-of-art LASSO solvers on an adaptive filter scenario.

Observation 1. *Given a sequence of batches,* $\mathcal{X} = \{X_t\}_{t=0}^{m}$, $\mathcal{Y} = \{\boldsymbol{y}_t\}_{t=0}^{m}$, *and a forgetting parameter* $r \in [0, 1]$, *if we define the sets of batches* $(\mathcal{X}_t, \mathcal{Y}_t)$ *as follows:*

$$(\mathcal{X}_t, \mathcal{Y}_t) = \begin{cases} \left(\begin{pmatrix} X_t \\ \sqrt{r} \cdot \mathcal{X}_{t-1} \end{pmatrix}, \begin{pmatrix} \boldsymbol{y}_t \\ \sqrt{r} \cdot \mathcal{Y}_{t-1} \end{pmatrix} \right) & \text{if } t > 0 \\ (X_t, \boldsymbol{y}_t) & \text{if } t = 0 \end{cases}$$

then, for $\boldsymbol{w} = \{r^{m-t}\}_{t=0}^{m}$, $f_{\mu}^{\mathcal{X}_m, \mathcal{Y}_m}(\theta) = l_{\mu, \boldsymbol{w}}^{\mathcal{X}, \mathcal{Y}}(\theta)$ *for all* $\theta \in \mathbb{R}^d$.

Proof Observe that the following chain of equality holds for all $\theta \in \mathbb{R}^d$:

$$\begin{aligned} f_{\mu}^{\mathcal{X}_m, \mathcal{Y}_m}(\theta) &= \frac{1}{2} \cdot \|\mathcal{X}_m\theta - \mathcal{Y}_m\|_2^2 + \mu \cdot \|\theta\|_1 \\ &= \frac{1}{2} \cdot (\|X_m\theta - \mathbf{y}_m\|_2^2 + r \cdot (\|\mathcal{X}_{m-1}\theta - \mathcal{Y}_{m-1}\|_2^2)) + \mu \cdot \|\theta\|_1 \\ &= \frac{1}{2} \cdot (\|X_m\theta - \mathbf{y}_m\|_2^2 + r \cdot (\|X_{m-1}\theta - \mathbf{y}_{m-1}\|_2^2)) + r^2 \cdot (\|\mathcal{X}_{m-2}\theta - \mathcal{Y}_{m-2}\|_2^2)) + \mu \cdot \|\theta\|_1 \\ &= \frac{1}{2} \cdot (\|X_m\theta - \mathbf{y}_m\|_2^2 + r \cdot (\|X_{m-1}\theta - \mathbf{y}_{m-1}\|_2^2)) + \ldots + r^m \cdot (\|X_0\theta - \mathbf{y}_0\|_2^2)) + \mu \cdot \|\theta\|_1 \\ &= \frac{1}{2} \cdot \sum_{t=0}^{m} r^{m-t} \cdot \|X_t\theta - \mathbf{y}_t\|_2^2 + \mu \cdot \|\theta\|_1 \\ &= l_{\mu, \mathbf{w}}^{\mathcal{X}, \mathcal{Y}}(\theta) \end{aligned}$$

□

The second theoretical result presented, Theorem 1, is a key property used in the construction of our homotopy-based solver. As we commented throughout Section 2, if one knows the correct set of active variables and corresponding signs, we can compute in closed form the solution Eq.4. Theorem 1 shows that both elements needed remain invariant for a certain interval within $[0, 1]$. Hence, we can easily compute such solutions in it.

Theorem 1. *If* $\|\widetilde{\widetilde{\boldsymbol{v}(s^*)}}\|_\infty < 1$ *is satisfied for a given* $s^* \in [0,1]$*, then there is* $\varepsilon > 0$*, such that* $\widetilde{\boldsymbol{v}(s)} = \widetilde{\boldsymbol{v}(s^*)}$ *and* $\|\widetilde{\widetilde{\boldsymbol{v}(s)}}\|_\infty < 1$*, for all* $s \in [s^*, s^* + \varepsilon)$*.*

Proof We will show that, for a sufficiently small $\varepsilon > 0$, there is a solution to Eq.4 for which $\mathcal{A}(s) = \mathcal{A}(s^*)$ and $\widetilde{\mathbf{v}(s)} = \widetilde{\mathbf{v}(s^*)}$, for all $s \in [s^*, s^* + \varepsilon)$.

First of all, since we are assuming $\mathcal{A}(s) = \mathcal{A}(s^*)$, we have that $\widetilde{X^s_m} = \widetilde{X^{s^*}_m}$, $\widetilde{\widetilde{X^s_m}} = \widetilde{\widetilde{X^{s^*}_m}}$, $\widetilde{\mathcal{X}^s_{m-1}} = \widetilde{\mathcal{X}^{s^*}_{m-1}}$, $\widetilde{\widetilde{\mathcal{X}^s_{m-1}}} = \widetilde{\widetilde{\mathcal{X}^{s^*}_{m-1}}}$, for all $s \in [s^*, s^* + \varepsilon)$. Moreover, we define $A = s^{*2} \cdot \widetilde{X^{s^*}_m}^T \widetilde{X^{s^*}_m} + r \cdot \widetilde{\mathcal{X}^{s^*}_{m-1}}^T \widetilde{\mathcal{X}^{s^*}_{m-1}}$ and $\mathbf{b} = s^{*2} \cdot \widetilde{X^{s^*}_m}^T \mathbf{y}_m + r \cdot \widetilde{\mathcal{X}^{s^*}_{m-1}}^T \mathcal{Y}_{m-1} - (r + s^* \cdot (1-r)) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)})$, which are known in advanced since they satisfy $\widetilde{\varphi(s^*)} = A^{-1}\mathbf{b}$. Taking this into consideration and re-writting the optimality conditions, Eq.6, using the binomial inverse theorem, for $\Lambda = \widetilde{X^{s^*}_m}^T \widetilde{X^{s^*}_m} A^{-1}$, we have

$$\begin{aligned}
\widetilde{\varphi(s)} &= [s^2 \cdot \widetilde{X^{s^*}_m}^T \widetilde{X^{s^*}_m} + r \cdot \widetilde{\mathcal{X}^{s^*}_{m-1}}^T \widetilde{\mathcal{X}^{s^*}_{m-1}}]^{-1} [s^2 \cdot \widetilde{X^{s^*}_m}^T \mathbf{y}_m + r \cdot \widetilde{\mathcal{X}^{s^*}_{m-1}}^T \mathcal{Y}_{m-1} - \\
&\quad -(r + s \cdot (1-r)) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)}] \\
&= [(s^2 - s^{*2}) \cdot \widetilde{X^{s^*}_m}^T \widetilde{X^{s^*}_m} + A]^{-1} [(s^2 - s^{*2}) \cdot \widetilde{X^{s^*}_m}^T \mathbf{y}_m + (s - s^*) \cdot (1-r) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)} + \mathbf{b}] \\
&= \widetilde{\varphi(s^*)} - (s - s^*) \cdot A^{-1} [\mathbb{I} + (s^2 - s^{2*}) \cdot \Lambda]^{-1} [(s + s^*) \cdot \widetilde{X^{s^*}_m}^T \widetilde{e} + (1-r) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)}], \quad (14)
\end{aligned}$$

where $\widetilde{e} = \widetilde{X^{s^*}_m}\widetilde{\varphi(s^*)} - \mathbf{y}_m$. Observe that $\widetilde{\varphi(s)}$ is a continuous function of s. Hence, since, $\widetilde{\varphi(s^*)}$ has no null entry by definition, then there exists $\varepsilon_1 > 0$ such that all entries of $\widetilde{\varphi(s)}$ remain not null for $s \in [s^*, s^* + \varepsilon_1)$.

Analogously, since $\widetilde{\widetilde{\mathbf{v}(s)}} = \frac{-1}{(r + s\cdot(1-r))\cdot\mu} \cdot (s^2 \cdot \widetilde{\widetilde{X^s_m}}^T (\widetilde{X^s_m \varphi(s)} - \mathbf{y}_m) + r \cdot \widetilde{\widetilde{\mathcal{X}^s_{m-1}}}^T (\widetilde{\mathcal{X}^s_{m-1}\varphi(s)} - \mathcal{Y}_{m-1}))$, and $s \in [s^*, s^* + \varepsilon) \subseteq [0,1]$, then $\widetilde{\widetilde{\mathbf{v}(s)}}$ is also a continuous function of s and, since $\|\widetilde{\widetilde{\mathbf{v}(s^*)}}\|_\infty < 1$, then there exists $\varepsilon_2 > 0$ such that $\|\widetilde{\widetilde{\mathbf{v}(s)}}\|_\infty < 1$ for $s \in [s^*, s^* + \varepsilon_2)$. Finally, take $\varepsilon_{s^*} = \min\{\varepsilon_1, \varepsilon_2\}$.

On the same token, Corollary 1, is a first attempt to compute the transition points -for the active variables that get annulled- using the continuity of Eq.4 within the interval presented in Theorem 1. □

Corollary 1. *Given a solution* $\widetilde{\varphi(s^*)} = A^{-1}\boldsymbol{b}$*, for* $s^* \in [0,1)$*, where* A *and* $\boldsymbol{b}$ *are defined by Eq.6, and* $\Lambda = \widetilde{X^{s^*}_m}^T \widetilde{X^{s^*}_m} A^{-1}$*, then*

$$\begin{aligned}
\widetilde{\varphi(s)} = \widetilde{\varphi(s^*)} - (s - s^*) \cdot A^{-1} [\mathbb{I} + (s^2 - s^{*2}) \cdot \Lambda]^{-1} [(s + s^*) \cdot \widetilde{X^{s^*}_m}^T \widetilde{e} + \\
+ (1-r) \cdot \mu \cdot \widetilde{\boldsymbol{v}(s^*)}], \quad (8)
\end{aligned}$$

where $\widetilde{e} = \widetilde{X^{s^*}_m}\widetilde{\varphi(s^*)} - \boldsymbol{y}_m$*, for all* $s \in [s^*, s^* + \varepsilon)$*, and* ε *defined in Theorem 1.*

Proof Taking $A = s^{*2} \cdot \widetilde{X_m^{s^*}}^T \widetilde{X_m^{s^*}} + r \cdot \widetilde{\mathcal{X}_{m-1}^{s^*}}^T \widetilde{\mathcal{X}_{m-1}^{s^*}}$ and $\mathbf{b} = s^{*2} \cdot \widetilde{X_m^{s^*}}^T \mathbf{y}_m + r \cdot \widetilde{\mathcal{X}_{m-1}^{s^*}}^T \mathcal{Y}_{m-1} - (r + s^* \cdot (1-r)) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)})$, the result is deduced in the chain of equalities of Eq.14. □

Due to the complexity of computing explicitly the transition points from the result presented in Corollary 1, in Observation 2, we introduce a condition that eases such a computation considering the eigenvalues of the Λ matrix.

Observation 2. *If the eigenvalues of the matrix* $(s^2 - s^{*2}) \cdot \Lambda$ *satisfy* $\max \lambda_{(s^2 - s^{*2}) \cdot \Lambda} < 1$*, then we can determine, for each variable* $j \in \mathcal{A}(s^*)$*, the value* s *for which* $\widetilde{\varphi(s)}_j = 0$*, by solving*

$$\sum_{i=0}^{\infty} (s^{*2} - s^2)^{i+1} \cdot \Gamma_{1j}^i - (s^{*2} - s^2)^i \cdot (s^* - s) \cdot \Gamma_{2j}^i = -\widetilde{\varphi(s^*)}_j, \tag{9}$$

where $\Gamma_1^i = A^{-1} \Lambda^i \widetilde{X_m^{s^*}}^T \widetilde{e}$ *and* $\Gamma_2^i = (r-1) \cdot \mu \cdot A^{-1} \Lambda^i \widetilde{X_m^{s^*}}^T \widetilde{\boldsymbol{v}(s^*)}$.

Proof Since $\max \lambda_{-}(s\hat{}2 - s\hat{}{-}2 * '') \cdot \Lambda'' < 1$, then we can apply the binomial series expansion over $[\mathbb{I} + (s^2 - s^{2*}) \cdot \Lambda]^{-1}$:

$$[\mathbb{I} + (s^2 - s^{2*}) \cdot \Lambda]^{-1} = \sum_{i=0}^{\infty} (-1)^i \cdot (s^2 - s^{2*})^i \cdot \Lambda^i \tag{15}$$

The result is then achieved by plugging Eq.15 into Eq.8.

□

Finally, in Theorem 2, we comment on the case proposed in [6], in which one observation is received at a time. In this scenario, we can easily compute the corresponding transition points explicitly from the data. The result present in Theorem 2 is a generalization of that obtained by [6], in which no adaptive filter is considered.

Theorem 2. *Given a solution* $\widetilde{\varphi(s^*)} = A^{-1}\boldsymbol{b}$*, for* $s^* \in [0, 1)$*, where* A *and* $\boldsymbol{b}$ *are defined by Eq.6. If* $n_m = 1$*,* $\boldsymbol{u} = A^{-1} \widetilde{X_m^{s^*}}^T$*,* $\alpha = \widetilde{X_m^{s^*}} \boldsymbol{u}$ *and* $\boldsymbol{z} = (r-1) \cdot \mu \cdot A^{-1} \widetilde{\boldsymbol{v}(s^*)}$*, then* $\widetilde{\varphi(s)}_j = 0$*, for*

$$s = \frac{\boldsymbol{z}_j \cdot \boldsymbol{m}_j}{2} \pm \left(s^{*2} - \varphi(s^*)_j \cdot \boldsymbol{m}_j + \boldsymbol{z}_j \cdot \boldsymbol{m}_j \cdot (\frac{\boldsymbol{m}_j}{4} + s^*) \right)^{\frac{1}{2}},$$

where $\boldsymbol{m}_j = (\alpha \cdot \varphi(s^*)_j - \widetilde{e} \cdot \boldsymbol{u}_j)^{-1}$*, for all* $j \in \mathcal{A}(s^*)$.

Proof First of all, observe that using Sherman-Morrison formula, for all $s \in [s^*, s^* + \varepsilon)$, we have.

$$\widetilde{\varphi(s)} = [s^2 \cdot \widetilde{X_m^{s^*}}^T \widetilde{X_m^{s^*}} + r \cdot \widetilde{\mathcal{X}_{m-1}^{s^*}}^T \widetilde{\mathcal{X}_{m-1}^{s^*}}]^{-1} [s^2 \cdot \widetilde{X_m^{s^*}}^T \mathbf{y}_m + r \cdot \widetilde{\mathcal{X}_{m-1}^{s^*}}^T \mathcal{Y}_{m-1} -$$

$$
\begin{aligned}
&-(r + s \cdot (1-r)) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)}] \\
&= [(s^2 - s^{*2}) \cdot \widetilde{X_m^{s^*}}^T \widetilde{X_m^{s^*}} + A]^{-1}](s^2 - s^{*2}) \cdot \widetilde{X_m^{s^*}}^T \mathbf{y}_m + (s - s^*) \cdot (1-r) \cdot \mu \cdot \widetilde{\mathbf{v}(s^*)} + \mathbf{b}] \\
&= \widetilde{\varphi(s^*)} - \frac{1}{1 + (s^2 - s^{*2}) \cdot \alpha} \cdot [(s^2 - s^{*2}) \cdot \widetilde{e} \cdot \mathbf{u} - (s - s^*) \cdot \mathbf{z}],
\end{aligned}
$$

where $\widetilde{e} = \widetilde{X_m^{s^*}} \widetilde{\varphi(s^*)} - \mathbf{y}_m$, $\mathbf{u} = A^{-1} \widetilde{X_m^{s^*}}^T$, $\alpha = \widetilde{X_m^{s^*}} \mathbf{u}$ and $\mathbf{z} = (r-1) \cdot \mu \cdot A^{-1} \widetilde{\mathbf{v}(s^*)}$. Hence, making use of Eq.16, we can compute explicitly the transition point for each $j \in \mathcal{A}(s^*)$, ς, i.e., $\widetilde{\varphi(s)}_j = 0$:

$$
\begin{aligned}
s \;=\; & -\frac{\mathbf{z}_j}{2 \cdot (\alpha \cdot \varphi(s^*)_j - \widetilde{e} \cdot \mathbf{u}_j)} \pm \Big(s^{*2} - (\alpha - \frac{\widetilde{e} \cdot \mathbf{u}_j}{\varphi(s^*)_j})^{-1} + \\
& + \frac{\mathbf{z}_j}{\alpha \cdot \varphi(s^*)_j - \widetilde{e} \cdot \mathbf{u}_j} \cdot (\frac{1}{4 \cdot (\alpha \cdot \varphi(s^*)_j - \widetilde{e} \cdot \mathbf{u}_j)} + s^*) \Big)^{\frac{1}{2}} \\
=\; & \frac{1}{2} \cdot \mathbf{z}_j \cdot \mathbf{m}_j \pm \Big(s^{*2} - \varphi(s^*)_j \cdot \mathbf{m}_j + \mathbf{z}_j \cdot \mathbf{m}_j \cdot (\frac{1}{4} \cdot \mathbf{m}_j + s^*) \Big)^{\frac{1}{2}},
\end{aligned}
$$

where $\mathbf{m}_j = (\alpha \cdot \varphi(s^*)_j - \widetilde{e} \cdot \mathbf{u}_j)^{-1}$. □

B Experimental Results

In this section, we further elaborate on the empirical results discussed in Section 3.

First of all, we would like to comment that, as described throughout the article, **AFLH** is a solver for LASSO with adaptive filter. **AFLH**[3] is able to process a large batch of new data while also implicitly applying a forgetting factor to the previously analyzed information to improve its predictability on non-stationary scenarios. In the case of the homotopy presented in **OOLH** this task is not explicitly possible. In order to empirically to compare **AFLH** to **OOLH**, in such cases, we first need to determine $\theta_{\mu/r}^{\mathcal{X}_{m-1}, \mathcal{Y}_{m-1}}$ via $\theta_{\mu}^{\mathcal{X}_{m-1}, \mathcal{Y}_{m-1}}$, which can be done via LARS, for instance. Afterwards, we apply the homotopy presented in [6] sequentially over each instance of $(X_m, \mathbf{y}_m)$.

In Tables 3-7, we complement the results presented over the artificial data sets, see Figs.2-6. In particular, we comment on the average relative computational running times with respect to all the different parameter values.

[3] An implementation of **AFLH** can be found at https://..

Table 3: Average relative computational running time for the different dimensionalities of the data, d.

Method	$d = 100$	$d = 1000$	$d = 10000$
AFLH	1.0×10^0	1.0×10^0	1.0×10^0
CD	4.4×10^2	6.2×10^3	4.0×10^5
CD_ws	2.9×10^2	1.2×10^3	7.9×10^3
LARS	4.7×10^2	1.4×10^3	8.6×10^3
OOLH	1.6×10^5	5.8×10^4	3.1×10^5

Table 4: Average relative computational running time for the different number of instances, n.

Method	$n = 500$	$n = 5000$	$n = 50000$
AFLH	1.0×10^0	1.0×10^0	1.0×10^0
CD	6.5×10^3	3.6×10^4	3.2×10^3
CD_ws	1.9×10^2	6.6×10^3	2.6×10^3
LARS	1.1×10^2	7.2×10^3	3.1×10^3
OOLH	2.6×10^2	9.3×10^3	4.4×10^5

Table 5: Average relative computational running time for the different batch sizes, n_0.

Method	$n_0 = 0.01$	$n_0 = 0.05$	$n_0 = 0.10$
AFLH	1.0×10^0	1.0×10^0	1.0×10^0
CD	1.5×10^4	1.6×10^4	1.4×10^4
CD_ws	3.2×10^3	3.1×10^3	2.9×10^3
LARS	3.6×10^3	3.5×10^3	3.2×10^3
OOLH	3.9×10^4	1.7×10^5	3.1×10^5

Table 6: Average relative computational running time for the different forgetting terms, r.

Method	$r = 0.50$	$r = 0.75$	$r = 1$
AFLH	2.1×10^0	1.0×10^0	1.0×10^0
CD	1.4×10^4	1.6×10^4	1.5×10^4
CD_ws	2.8×10^3	3.4×10^3	3.2×10^3
LARS	2.9×10^3	3.7×10^3	3.6×10^3
OOLH	1.8×10^5	1.7×10^5	1.8×10^5

Table 7: Average relative computational running time for the different penalty terms, μ.

Method	$\mu = 0.01$	$\mu = 0.05$	$\mu = 0.10$
AFLH	1.0×10^0	1.0×10^0	1.0×10^0
CD	3.0×10^4	9.4×10^3	5.8×10^3
CD_ws	3.1×10^3	3.4×10^3	2.9×10^3
LARS	3.5×10^3	3.6×10^3	3.2×10^3
OOLH	1.8×10^5	1.9×10^5	1.6×10^5

In Tables 2-7, we observe again that in general the dimensionality of the problem seems to be the main factor affecting the overall performance of the methods, except for **OOLH**, which is also deeply affected by the size of the batches and the size of the data sets, n_0 and n, respectively. The previously discussed results are obtained for the case $m = 1$. Observe that, for $m > 1$, if more batches arrive in the stream of data, then the difference of computational time is expected to be even more favourable for **AFLH** in comparison to the other methods.

Finally, in order to verify that all the considered algorithms indeed converged to the expected solution of the problem, in Figs.7-11, we show the relative error of the LASSO with adaptive filtering error function, Eq.2, of the

optimal solution with respect to the solution obtained for each method, for all the parameters.

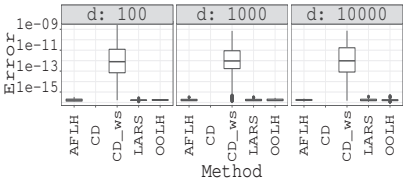

Figure 7: Relative error w.r.t. the dimensionality of the problem, d.

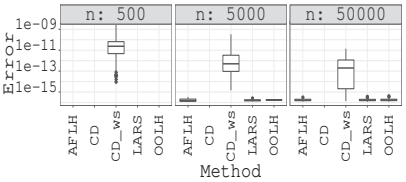

Figure 8: Relative error w.r.t. the number of instances, n.

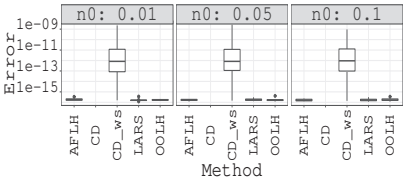

Figure 9: Relative error w.r.t. batch size, n_0.

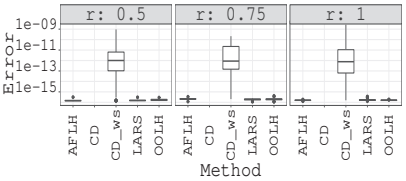

Figure 10: Relative computational error w.r.t. the forgetting term, r.

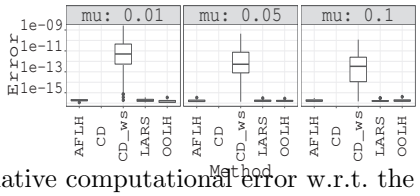

Figure 11: Relative computational error w.r.t. the penalty term, μ.

B.1 Additional Results on Real Data Sets

In this section, we comment on additional results that were obtained on real regression data sets obtained from the `UCI repository`, see Table 8. As in Section 3, we consider the following parameter values: $n_0 \in \{0.01, 0.05, 0.10\}$, $r \in \{0.50, 0.75, 1\}$ and $\mu \in \{0.001 \cdot \mu_{\max}, 0.005 \cdot \mu_{\max}, 0.01 \cdot \mu_{\max}, 0.05 \cdot \mu_{\max}, 0.10 \cdot \mu_{\max}\}$.

Table 8: Data Sets Information.

Data Set	n	d
Madelon NIPS (**MN**)	1800	499
Isolet (**IS**)	6238	617
Superconductivity (**SC**)	21263	81
Song (**SG**)	515345	90

In Figs 12-15, we observe the relative running times w.r.t. the algorithm that converged the fastest for the different parameters of the problem and data sets considered:

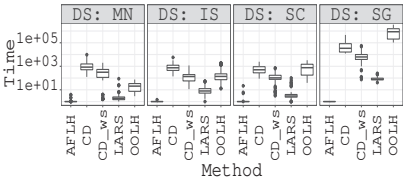

Figure 12: Relative computational time w.r.t. the different data sets.

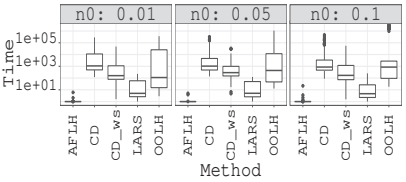

Figure 13: Relative computational time w.r.t. batch size, n_0.

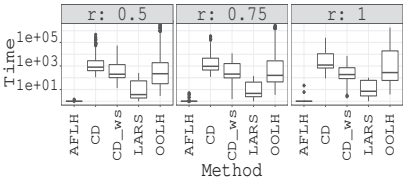

Figure 14: Relative computational time w.r.t. the forgetting term, r.

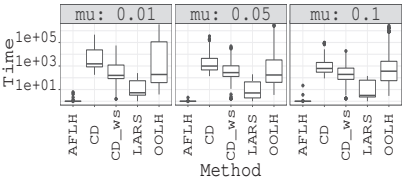

Figure 15: Relative computational time w.r.t. the penalty term, μ.

As observed in Section 3, **AFLH** converges the fastest regardless of the setting considered for the LASSO with adaptive filtering problem. The most competitive method besides **AFLH** is **LARS**, which best performance is achieved in the smallest data set, **MN**. However in this case, **LARS** is still 3.09 times slower than **AFHL**, on average. On the other hand, for the largest data set, **SG**, we observe a remarkable running time difference between **AFHL** and its competitors: **CD**, **CD_ws**, **LARS** and **OOLH** take 6.9×10^4, 8.8×10^3, 8.9×10^1 and 9.6×10^5 more running time than **AFHL**. This is expected, due to the large size of the batch being processed.

References

[1] Yuan L, Liu J, Ye J. Efficient methods for overlapping group lasso. In: Advances in neural information processing systems; 2011. p. 352–360.

[2] Hastie T, Tibshirani R, Wainwright M. Statistical learning with sparsity: the lasso and generalizations. CRC press; 2015.

[3] Monti RP, Anagnostopoulos C, Montana G. A framework for adaptive regularization in streaming Lasso models. stat. 2016;1050:28.

[4] Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological). 1996;58(1):267–288.

[5] Chen SS, Donoho DL, Saunders MA. Atomic decomposition by basis pursuit. SIAM review. 2001;43(1):129–159.

[6] Garrigues P, Ghaoui LE. An homotopy algorithm for the Lasso with online observations. In: Advances in neural information processing systems; 2009. p. 489–496.

[7] Tibshirani RJ, et al. The lasso problem and uniqueness. Electronic Journal of statistics. 2013;7:1456–1490.

[8] Efron B, Hastie T, Johnstone I, Tibshirani R, et al. Least angle regression. The Annals of statistics. 2004;32(2):407–499.

[9] Fu WJ. Penalized regressions: the bridge versus the lasso. Journal of computational and graphical statistics. 1998;7(3):397–416.

[10] Kim Y, Kim J. Gradient LASSO for feature selection. In: Proceedings of the twenty-first international conference on Machine learning; 2004. p. 60.

[11] Osborne MR, Presnell B, Turlach BA. A new approach to variable selection in least squares problems. IMA journal of numerical analysis. 2000;20(3):389–403.

[12] Lokhorst J, Venables B, Turlach B, Maechler M. lasso2: L1 constrained estimation aka 'lasso'. R package version. 2007;p. 1–2.

[13] Boyd S, Boyd SP, Vandenberghe L. Convex optimization. Cambridge university press; 2004.

[14] Nesterov YE, Nemirovski A. Interior-point polynomial methods in convex programming, vol. 14 of Stud. Appl Math, SIAM, Philadelphia. 1994;.

[15] Wright SJ. Primal-dual interior-point methods. SIAM; 1997.

[16] Ye Y. Interior point algorithms: theory and analysis. vol. 44. John Wiley & Sons; 2011.

[17] Kim SJ, Koh K, Lustig M, Boyd S, Gorinevsky D. An interior-point method for large-scale l_1-regularized least squares. IEEE journal of selected topics in signal processing. 2007;1(4):606–617.

[18] Daubechies I, Defrise M, De Mol C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences. 2004;57(11):1413–1457.

[19] Ida Y, Fujiwara Y, Kashima H. Fast Sparse Group Lasso. In: Advances in Neural Information Processing Systems; 2019. p. 1702–1710.

[20] Liu H, Palatucci M, Zhang J. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In: Proceedings of the 26th Annual International Conference on Machine Learning; 2009. p. 649–656.

[21] Qin Z, Scheinberg K, Goldfarb D. Efficient block-coordinate descent algorithms for the group lasso. Mathematical Programming Computation. 2013;5(2):143–169.

[22] Wu TT, Lange K, et al. Coordinate descent algorithms for lasso penalized regression. The Annals of Applied Statistics. 2008;2(1):224–244.

[23] Hastie T, Rosset S, Tibshirani R, Zhu J. The entire regularization path for the support vector machine. Journal of Machine Learning Research. 2004;5(Oct):1391–1415.

[24] Shor NZ. Minimization methods for non-differentiable functions. vol. 3. Springer Science & Business Media; 2012.

[25] Kim J, Kim Y, Kim Y. A gradient-based optimization algorithm for lasso. Journal of Computational and Graphical Statistics. 2008;17(4):994–1009.

[26] Langford J, Li L, Zhang T. Sparse online learning via truncated gradient. In: Advances in neural information processing systems; 2009. p. 905–912.

[27] Mairal J, Yu B. Complexity analysis of the lasso regularization path. arXiv preprint arXiv:12050079. 2012;.

[28] Malioutov DM, Cetin M, Willsky AS. Homotopy continuation for sparse signal representation. In: Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.. vol. 5. IEEE; 2005. p. v–733.

[29] Rosset S, Zhu J. Piecewise linear regularized solution paths. The Annals of Statistics. 2007;p. 1012–1030.

[30] Gärtner B, Jaggi M, Maria C. An exponential lower bound on the complexity of regularization paths. arXiv preprint arXiv:09034817. 2009;.

[31] Heard NA, Weston DJ, Platanioti K, Hand DJ, et al. Bayesian anomaly detection methods for social networks. The Annals of Applied Statistics. 2010;4(2):645–662.

[32] Weiskopf N. Real-time fMRI and its application to neurofeedback. Neuroimage. 2012;62(2):682–692.

[33] Gaber MM, Zaslavsky A, Krishnaswamy S. Mining data streams: a review. ACM Sigmod Record. 2005;34(2):18–26.

[34] Krempl G, Žliobaite I, Brzeziński D, Hüllermeier E, Last M, Lemaire V, et al. Open challenges for data stream mining research. ACM SIGKDD explorations newsletter. 2014;16(1):1–10.

[35] Aggarwal CC. Data streams: models and algorithms. vol. 31. Springer Science & Business Media; 2007.

[36] Hayes MH. Statistical digital signal processing and modeling. John Wiley & Sons; 2009.

[37] Haykin SS. Adaptive filter theory. Pearson Education India; 2008.

[38] Gustafsson F, Gustafsson F. Adaptive filtering and change detection. vol. 1. Citeseer; 2000.

[39] Monti RP, Anagnostopoulos C, Montana G. Adaptive regularization for lasso models in the context of nonstationary data streams. Statistical Analysis and Data Mining: The ASA Data Science Journal. 2018;11(5):237–247.