Received: 14 March 2018 | Revised: 7 June 2018 | Accepted: 15 June 2018

DOI: 10.1002/sam.11390

ORIGINAL ARTICLE

WILEY

Adaptive regularization for Lasso models in the context of nonstationary data streams

Ricardo P. Monti[1,2] | **Christoforos Anagnostopoulos**[1] | **Giovanni Montana**[3]

[1]Department of Mathematics, Imperial College London, London, UK
[2]Gatsby Computational Neuroscience Unit, UCL, London, UK
[3]Warwick Manufacturing Group, University of Wawrick, Coventry, UK

Correspondence
Ricardo P. Monti, Gatsby Computational Neuroscience Unit, UCL, London, W1T 4JG, UK.
Email: r.monti@ucl.ac.uk

Large-scale, streaming data sets are ubiquitous in modern machine learning. Streaming algorithms must be scalable, amenable to incremental training, and robust to the presence of nonstationarity. In this work we consider the problem of learning ℓ_1 regularized linear models in the context of streaming data. In particular, the focus of this work revolves around how to select the regularization parameter when data arrives sequentially and the underlying distribution is nonstationary (implying the choice of optimal regularization parameter is itself time-varying). We propose a framework through which to infer an adaptive regularization parameter. Our approach employs an ℓ_1 penalty constraint where the corresponding sparsity parameter is iteratively updated via stochastic gradient descent. This serves to reformulate the choice of regularization parameter in a principled framework for online learning. The proposed method is derived for linear regression and subsequently extended to generalized linear models. We validate our approach using simulated and real data sets, concluding with an application to a neuroimaging data set.

KEYWORDS

adaptive filtering, ℓ_1 regularization, nonstationary data streams, time-varying sparsity

1 | INTRODUCTION

We are interested in learning ℓ_1 regularized regression models in the context of streaming, nonstationary data. There has been significant research relating to the estimation of such models in a streaming data context [3,4]. However, a fundamental aspect that has been overlooked is the selection of the regularization parameter. The choice of this parameter dictates the severity of the regularization penalty. While the underlying optimization problem remains convex, distinct choices of such a parameter yield models with vastly different characteristics. This poses significant concerns from the perspective of model performance and interpretation. It, therefore, follows that selecting such a parameter is an important problem that must be addressed in a data-driven manner.

Many solutions have been proposed through which to select the regularization parameter in a nonstreaming context. For example, stability-based approaches have been proposed in the context of linear regression [18]. Other popular alternatives include cross-validation and information theoretic techniques [9]. However, in a streaming setting such approaches are infeasible due to the limited computational resources available. Moreover, the statistical properties of the data may vary over time; a common manifestation being concept drift [1]. This complicates the use of subsampling methods as the data can no longer be assumed to follow a stationary distribution. Furthermore, as we argue in this work, it is conceivable that the optimal choice of regularization parameter may itself vary over time. It is also important to note that traditional approaches such as change-point detection cannot be employed as there is no readily available pivotal quantity. It, therefore, follows that novel methodologies are required in order to tune regularization parameters in an online setting.

Applications involving streaming data sets are abundant, ranging from finance [8] to cyber-security [11] and neuroscience [13,14]. In this work we are motivated by the latter application, where penalized regression models are often employed to decode statistical dependencies across spatially

remote brain regions, referred to as functional connectivity [28]. A novel avenue for neuroscientific research involves the study of functional connectivity in real-time [22,30]. Such research faces challenges due to the nonstationary as well as potentially high dimensional nature of neuroimaging data [21,24]. In order to address these challenges, many of the proposed methods to date have employed fixed sparsity parameters. However, such choices are typically justified only by the methodological constraints associated with updating the regularization parameter, as opposed to for biological reasons.

In order to address these issues, we propose a framework through which to learn an adaptive sparsity parameter in an online fashion. The proposed framework, named real-time adaptive penalization (RAP), is capable of iteratively learning time-varying regularization parameters via the use of adaptive filtering. Briefly, adaptive filtering methods are semi-parametric methods which employ information from recent observations to tune a parameter of interest. In this manner, adaptive filtering methods are capable of handling temporal variation which cannot easily be modeled explicitly [10]. The contributions of this work can be summarized as follows:

1. We propose and validate a framework through which to tune a time-varying sparsity parameter for ℓ_1 regularized linear models in real-time.
2. The proposed framework is subsequently extended to the context regularized generalized linear models (GLMs).
3. An empirical validation is provided using both synthetic and real data sets together with an application to a neuroimaging data set.

The remainder of this manuscript is organized as follows. Related work is discussed in Section 2. We formally describe our problem in Section 3 and the proposed framework is introduced in Section 4. We provide empirical evidence based on real and simulated data in Section 5.

2 | RELATED WORK

Regularized methods have established themselves as popular and effective tools through which to handle high-dimensional data [9]. Such methods employ regularization penalties as a mechanism through which to constraint the set of candidate solutions, often with the goal of enforcing specific properties such as parsimony. In particular, ℓ_1 regularization is widely employed as a convex approximation to the combinatorial problem of model selection.

However, the introduction of an ℓ_1 penalty requires the specification of the associated regularization parameter. The task of tuning such a parameter has primarily been studied in the context of nonstreaming, stationary data. Stability selection procedures, introduced by ref. [18], effectively look to bypass the selection of a specific regularization parameter by instead fitting multiple models across subsampled data. Variables are subsequently selected according to the proportion of all models in which they are present. In this manner, stability selection is able to provide important theoretical guarantees, albeit while incurring an additional computational burden. Other popular approaches involve the use of cross-validation or information theoretic techniques. However, such methods cannot be easily adapted to handle streaming data.

Online learning with the ℓ_1 constraints has also been studied extensively and many computationally efficient algorithms are available. A stochastic gradient descent algorithm is proposed by ref. [3]. More generally, online learning of regularized objective functions has been studied extensively by ref. [4] who propose a general class of computationally efficient methods based on proximal gradient descent. The aforementioned methods all constitute important advances in the study of sparse online learning algorithms. However, a fundamental issue that has been overlooked corresponds to the selection of the regularization parameters. As such, current methodologies are rooted on the assumption that the regularization parameter remains fixed. It follows that the regularization parameter may itself vary over time, yet selecting such a parameter in a principled manner is nontrivial. The focus of this work is to present and validate a framework through which to automatically select and update the regularization parameter in real-time. The framework presented in this work is therefore complementary and can be employed in conjunction with many of the preceding techniques. In a similar spirit to the methods proposed in this manuscript, ref. [7] propose a method for selecting the regularization parameter in the context of sequential data but do not consider nonstationary data, which is the explicit focus of this work. We further consider the extension to general linear models, leading to a wider range of potential applications.

More generally, the automatic selection of hyper-parameters has recently become an active topic in machine learning [27]. Interest in this topic has been catalyzed by the success of deep learning algorithms, which typically involve many such hyper-parameters. Sequential model-based optimization (SMBO) methods such as Bayesian optimization employ a probabilistic surrogate to model the generalization performance of learning algorithms as samples from a Gaussian process [27], leading to expert-level performance in many cases. It follows that such methods may be employed to tune regularization parameters in the context of penalized linear regression models. However, there are several important differences between the SMBO framework and the proposed framework. The most significant difference relates to the fact that the proposed framework employs gradient information in order to tune the regularization parameter while SMBO methods such as Bayesian optimization are rooted in the use of a probabilistic surrogate model. This allows the SMBO framework to be applied in a wide range of settings while the proposed framework focuses exclusively on Lasso regression models.

However, as we describe in this work, the use of gradient information makes the RAP framework ideally suited in the context of nonstationary, streaming data. This is in contrast to SMBO techniques, which typically assume the data is stationary.

3 | PRELIMINARIES

In this section we introduce the necessary ingredients to derive the proposed framework. We begin formally defining the problem addressed in this work in Section 3.1. Adaptive filtering methods are introduced in Section 3.2.

3.1 | Problem setup

In this work we are interested in streaming data problems. Here it is assumed that pairs (X_t, y_t) arrive sequentially over time, where $X_t \in \mathbb{R}^{p \times 1}$ corresponds to a p-dimensional vector of predictor variables and y_t is a univariate response. The objective of this work is to learn time-varying linear regression models* to accurately predict future responses, y_{t+1}, from predictors, X_{t+1}. An ℓ_1 penalty, parameterized by $\lambda \in \mathbb{R}_+$, is introduced in order to encourage sparse solutions as well as to ensure the problem is well-posed from an optimization perspective. This corresponds to the Lasso model introduced by ref. [29]. For a given choice of regularization parameter, λ, time-varying regression coefficients can be estimated by minimizing the following convex objective function:

$$L_t(\beta, \lambda) = \sum_{i=1}^{t} w_i (y_i - X_i^T \beta)^2 + \lambda \|\beta\|_1, \tag{1}$$

where $w_i > 0$ are weights indicating the importance given to past observations [1]. Typically, w_i decay monotonically in a manner which is proportional to the chronological proximity of the ith observation. For example, weights w_i may be tuned using a fixed forgetting factor or a sliding window. We refer readers to ref. [10] for a detailed overview of sliding window and forgetting factor methods.

In a nonstationary context, the optimal estimates of regression parameters, $\widehat{\beta}_t$, may vary over time. The same argument can be posed in terms of the selected regularization parameter, λ. For example, this may arise due to changes in the underlying sparsity or changes in the signal-to-noise ratio. While there exists a wide range of methodologies through which to update regression coefficients in a streaming fashion, the choice of regularization parameter has been largely ignored. As such, the primary objective of this work is to propose a framework through which to learn time-varying regularization parameter in real-time. The proposed framework seeks to iteratively update the regularization parameter via stochastic gradient descent and is therefore conceptually related to adaptive filtering theory [10], which we introduce below.

*We note that the proposed framework will be extended to GLMs in Section 4.3. For clarity we first formulate our approach in the context of linear regression.

3.2 | Adaptive filtering

Filtering, as defined in ref. [10], is the process through which information regarding a quantity of interest is assimilated using data measured up to and including time t. In many real-time applications, the quantity of interest is assumed to vary over time. The task of a filter therefore corresponds to effectively controlling the rate at which past information is discarded. Adaptive filtering methods provide an elegant method through which to handle a wide range of nonstationary behavior without having to explicitly model the dynamic properties of the data stream.

The simplest filtering methods discard information at a constant rate, for example, determined by a fixed forgetting factor. More sophisticated methods are able to exploit gradient information to determine the aforementioned rate. Such methods are said to be adaptive as the rate at which information is discarded varies over time. It follows that the benefits of adaptive methods are particularly notable in scenarios where the quantity of interest is highly nonstationary.

To further motivate discussion, we briefly review filtering in the context of fixed forgetting factors for streaming linear regression. In such a scenario, it suffices to store summary statistics for the mean and sample covariance. For a given fixed forgetting factor $r \in (0, 1]$, the sample mean can be recursively estimated as follows:

$$\overline{X}_t = \left(1 - \frac{1}{\omega_t}\right) \overline{X}_{t-1} + \frac{1}{\omega_t} X_t, \tag{2}$$

where ω_t is a normalizing constant defined as:

$$\omega_t = \sum_{i=1}^{t} r^{t-i} = r \cdot \omega_{t-1} + 1. \tag{3}$$

Similarly, the sample covariance can be learned iteratively:

$$S_t = \left(1 - \frac{1}{\omega_t}\right) S_{t-1} + \frac{1}{\omega_t} (X_t - \overline{X}_t)^T (X_t - \overline{X}_t). \tag{4}$$

It is clear that the value of r directly determines the adaptivity of a filter as well as its susceptibility to noise. However, in many practical scenarios the choice of r presents a challenge as it assumes some knowledge about the *degree* of nonstationarity of the system being modeled as well as an implicit assumption that this is constant [10]. Adaptive filtering methods address these issues by allowing r to be tuned online in a data-driven manner. This is achieved by quantifying the performance of current parameter estimates for new observations, X_{t+1}. Throughout this work we denote such a measure by $C(X_{t+1})$.

A popular approach is to define $C(X_{t+1})$ to be the residual error on unseen data [10]. In such a setting, it is clear that $C(X_t)$ depends on current estimates of sufficient statistics such as S_t, and is therefore a function of the forgetting factor, r.

Then assuming $\frac{\partial C(X_{t+1})}{\partial r}$ can be efficiently calculated, our parameter of interest can be updated in a stochastic gradient descent framework:

$$r_{t+1} = r_t - \varepsilon \frac{\partial C(X_{t+1})}{\partial r}\bigg|_{r=r_t} \tag{5}$$

where ε is a small step-size parameter which determines the learning rate. The objective of this work therefore corresponds to extending adaptive filtering methods to the domain of learning a time-varying regularization parameter for Lasso regression models.

4 | METHODS

As noted previously, the choice of parameter λ dictates the severity of the regularization penalty. Different choices of λ result in vastly different estimated models. While several data-driven approaches are available for selecting λ in an offline setting, such methods are typically not feasible for streaming data for two reasons. First, limited computational resources pose a practical restriction. Second, data streams are often nonstationary and rarely satisfy *iid* assumptions required for methods based on the bootstrap [1]. Moreover, it is important to note that traditional methods such as change point detection cannot be employed due to the absence of a readily available pivotal quantity for λ.

We begin by outlining the RAP framework in the linear regression setting in Section 4.1. Section 4.2 outlines the resulting algorithm and computational considerations. In Section 4.3 we extend the RAP framework to the setting of GLMs.

4.1 | Proposed framework

We propose to learn a time-varying sparsity parameter in an adaptive filtering framework. This allows the proposed method to relegate the choice of sparsity parameter to the data. Moreover, by allowing λ_t to vary over time the proposed method is able to naturally accommodate data sets where the underlying sparsity may be nonstationary.

We define the empirical objective to be the look-ahead negative log-likelihood, defined as:

$$C_{t+1} = C(X_{t+1}, y_{t+1}) = \|y_{t+1} - X_{t+1}\widehat{\beta}_t(\lambda_t)\|_2^2, \tag{6}$$

where we write $\widehat{\beta}_t(\lambda_t)$ to emphasize the dependence of the estimated regression coefficients on the current value of the regularization parameter, λ_t. Following Section 3.2, the regularization parameter can be iteratively updated as follows:

$$\lambda_{t+1} = G(\lambda_t) = \lambda_t - \varepsilon \frac{\partial C_{t+1}}{\partial \lambda_t}. \tag{7}$$

We note that for convenience we write $\frac{\partial C_{t+1}}{\partial \lambda_t}$ to denote the derivate of C_{t+1} with respect to λ evaluated at $\lambda = \lambda_t$ (i.e, $\frac{\partial C_{t+1}}{\partial \lambda}\Big|_{\lambda=\lambda_t}$). We note that λ_t is bounded below by zero, in which case no regularization is applied, and above by $\lambda_t^{\max} = \max_j \left\{ \left| \sum_{i=1}^t w_i y_i X_{i,j} \right| \right\}$, in which case all regression coefficients are zero [6].

The proposed framework requires only the specification of an initial sparsity parameter, λ_0, together with a stepsize parameter, ε. In this manner the proposed framework effectively replaces a fixed sparsity parameter with a stepsize parameter, ε. This is desirable as the choice of a fixed sparsity parameter is difficult to justify in the context of streaming, nonstationary data. Moreover, any choice of λ is bound to be problem specific. In comparison, we are able to interpret ε as a stepsize parameter in a stochastic gradient descent scheme. As a result, there are clear guidelines which can be followed when selecting ε [2].

Once the regularization parameter has been updated, estimates for the corresponding regression coefficients can be obtained by minimizing $L_{t+1}(\beta, \lambda_{t+1})$, for which there is a wide literature available [3,4]. The challenge in this work therefore corresponds to efficiently calculating the derivative in Equation (7). Through the chain rule, this can be decomposed as:

$$\frac{\partial C_{t+1}}{\partial \lambda_t} = \frac{\partial C_{t+1}}{\partial \widehat{\beta}_t} \cdot \frac{\partial \widehat{\beta}_t}{\partial \lambda_t}. \tag{8}$$

The first term in Equation (8) can be obtained by direct differentiation. In the case of the second term, we leverage the results of refs. ([5,26] who demonstrate that the Lasso solution path is piecewise linear as a function of λ. By implication, $\frac{\partial \widehat{\beta}_t}{\partial \lambda_t}$ must be piecewise constant. Furthermore, there is a simple, closed-form solution for $\frac{\partial \widehat{\beta}_t}{\partial \lambda_t}$.

Proposition 1. *[Adapted from ref*. [26]*] In the context of ℓ_1 penalized linear regression models, the derivative $\frac{\partial \widehat{\beta}_t}{\partial \lambda_t}$ is piecewise constant and can be obtained in closed form.*

Proof. For any choice of regularization parameter, λ, we write $\widehat{\beta}_t(\lambda)$ to denote the minimizer of Equation (1). Recall that the objective, $L_t(\beta, \lambda)$, is nonsmooth due to the presence of the ℓ_1 penalty. As a result, the subgradient of $L_t(\beta, \lambda)$ must satisfy:

$$\begin{aligned}\nabla_\beta (L_t(\beta, \lambda))|_{\beta=\widehat{\beta}_t(\lambda)} = -X_{1:t}^T \; W y_{1:t} + X_{1:t}^T W \; X_{1:t}\widehat{\beta}_t(\lambda) \\ + \lambda \, \text{sign}(\widehat{\beta}_t(\lambda)) \ni 0,\end{aligned} \tag{9}$$

where we W is a diagonal matrix with elements w_i and we write $X_{1:t}$ to denote a matrix where the ith row is X_i. It is important to note that Equation (9) holds for any choice of λ; however, the corresponding estimate of regression coefficients, $\widehat{\beta}_t(\lambda)$, will necessarily change. Further, taking the derivative with respect to the regularization parameter λ yields:

$$\begin{aligned}&\frac{\partial}{\partial \lambda}(\nabla_\beta L_t(\beta, \lambda)|_{\beta=\widehat{\beta}_t(\lambda)}) = 0 \\ &= \frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda} \nabla_\beta^2 L_t(\widehat{\beta}_t(\lambda), \lambda) + \text{sign}(\widehat{\beta}_t(\lambda)) \\ &= \frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda}(X_{1:t}^T W \; X_{1:t}) + \text{sign}(\widehat{\beta}_t(\lambda)).\end{aligned}$$

Rearranging yields:

$$\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda} = -(X_{1:t}^T W \; X_{1:t})^{-1} \operatorname{sign}(\widehat{\beta}_t(\lambda)) = -(S_t)^{-1} \operatorname{sign}(\widehat{\beta}_t(\lambda)). \quad (10)$$

□

From Proposition 1 we have that the derivative, $\frac{\partial C_{t+1}}{\partial \lambda_t}$, can be computed in closed form.

Moreover, we note that the derivative in Equation (10) is only nonzero over the active set of regression coefficients, $\mathscr{A}_t = \{i : (\widehat{\beta}_t(\lambda_t))_i \neq 0\}$, and zero elsewhere. In practice we must therefore consider two scenarios:

- The active set is nonempty (ie, $\mathscr{A}_t \neq \emptyset$). In this case Equation (10) is well-defined.
- The active set is empty. In this case we follow the LARS algorithm and take a step in the direction of the most correlated predictor: $\widehat{j} = \underset{j}{\operatorname{argmax}} \left\{ | \sum_{i=1}^t w_i y_i X_{i,j} | \right\}$. As such, we define the gradient as follows:

$$\left(\frac{\partial \widehat{\beta}(\lambda)}{\partial \lambda} \right)_k = \delta_{k,\widehat{j}} \operatorname{sign} \left(\left[\sum_{i=1}^t w_i y_i X_{i,k} \right] \right),$$

where δ is the dirac-delta function. As such, all entries of $\frac{\partial \widehat{\beta}(\lambda)}{\partial \lambda}$ will be zero except for the entry corresponding to the most correlated predictor.

4.2 | Streaming Lasso regression

At each iteration, a new pair (X_{t+1}, y_{t+1}) is received and employed to update both the time-varying regularization parameter, λ_t, as well as the corresponding estimate of regression coefficients, $\widehat{\beta}_t(\lambda_t)$. The former involves computing the derivative $\frac{\partial C_{t+1}}{\partial \lambda_1}$ as outlined in Section 4.1. The latter involves solving a convex optimization problem which can be addressed in a variety of ways. In this work we look to iteratively estimate regression coefficients using coordinate descent methods [6]. Such methods are easily amenable to streaming data and allow us to exploit previous estimates as warm starts. In our experience, the use of warm starts leads to convergence within a handful of iterations. Pseudo-code detailing the proposed RAP framework is given in Algorithm 1.

Algorithm 1: **R**eal-time **A**daptive **P**enalization

Require: $\epsilon \in \mathbb{R}_+$ and $r \in (0, 1]$

1: **for** $t \leftarrow 1 \dots t, \dots$ **do**
2: receive new (X_{t+1}, y_{t+1})
3: compute $\frac{\partial \hat{\beta}_t(\lambda)}{\partial \lambda_t}$ using equation (10)
4: set $\frac{\partial C_{t+1}}{\partial \lambda_t} = \frac{\partial C_{t+1}}{\partial \hat{\beta}_t(\lambda)} \frac{\partial \hat{\beta}_t(\lambda)}{\partial \lambda_t}$
5: update $\lambda_{t+1} = \lambda_t - \epsilon \frac{\partial C_{t+1}}{\partial \lambda_t}$
6: $\hat{\beta}_{t+1}(\lambda_{t+1}) = \underset{\beta}{\operatorname{argmin}} \{L_{t+1}(\beta, \lambda_{t+1})\}$

4.2.1 | Computational considerations

With respect to the computational and memory demands, the major expense incurred when calculating $\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda_t}$ involves inverting the sample covariance matrix. The need to compute and store the inverse of the sample covariance is undesirable in the context of high-dimensional data. As a result, the following approximation is also considered:

$$\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda_t} \approx -(\operatorname{diag}(S_t))^{-1} \operatorname{sign}(\widehat{\beta}_t(\lambda)). \quad (11)$$

Such approximations are frequently employed in streaming or large data applications [4]. The approximate update therefore has a time and memory complexity that is proportional to the cardinality of the active set, $\mathscr{A}_t$. In practice, we find that the empirical performance between the use of the exact and approximate gradients to be small (eg, Figure 2 and Table 1). This may be due to the fact that in the context of highly nonstationary data it is often difficult to obtain a reliable estimate of the sample covariance, S_t, implying that a diagonal approximate may suffice.

4.3 | Extension to GLMs

While the preceding sections focused on linear regression, we now extend the proposed framework to a wider class of GLM models. As such, we assume that observations y_t follow an exponential family distribution such that $[y_t] = \mu_t$ and $\operatorname{Var}(y_t) = V_t$. In the context of GLMs, it is assumed that a (potentially nonlinear) link function is employed to relate the mean, μ_t, to a linear combination of predictors:

$$\eta_t = g(\mu_t) = X_t^T \beta_{t-1}. \quad (12)$$

We note that when y_t is assumed to follow a Gaussian distribution we recover linear regression as described in Section 4.1. Conversely, if y_t follows a Binomial distribution we obtain streaming logistic regression. The log-likelihood of an observed response, y_t, can be expressed as ref. [15]:

$$l(y_t; \theta_t) = \frac{y_t \theta_t - b(\theta_t)}{a(\phi)} + c(y_t, \phi), \quad (13)$$

where $a(\cdot)$, $b(\cdot)$, and $c(\cdot)$ are functions which vary according to the distribution of the response and $\theta_t = \theta(\beta_t)$ is the corresponding canonical parameter. Throughout this work it is assumed that the dispersion parameter, ϕ, is known and fixed.

Analogously to Equation (1), we estimate ℓ_1 regularized regression coefficients by minimizing the reweighted negative log-likelihood objective:

$$L_t(\beta, \lambda) = -\sum_{i=1}^t w_i [y_i \, \theta(\beta_i) - b\{\theta(\beta_i)\}] + \lambda \|\beta\|_1, \quad (14)$$

where w_i are weights as before. In the remainder of this manuscript we focus on two popular cases, detailed below, but we note that the proposed framework can be employed in a much wider range of settings.

TABLE 1 Detailed results for simulation involving nonstationary data over $N = 500$ independent iterations. We report the mean negative log-likelihood, $\overline{C}_t$, as well as the mean F-score, $\overline{F}_t$. Standard errors are provided in brackets

	Linear regression		Logistic regression	
Algorithm	**A$\overline{C}_t$**	**$\overline{F}_t$**	**$\overline{C}_t$**	**$\overline{F}_t$**
Fixed (CV)	0.58 (0.05)	0.49 (0.05)	0.25 (0.07)	0.49 (0.04)
Fixed (SMBO)	0.63 (0.05)	0.50 (0.07)	0.26 (0.08)	0.49 (0.05)
Stepwise Oracle	**0.51** (0.04)	**0.56** (0.04)	**0.21** (0.06)	**0.53** (0.04)
RAP (No forget)	0.63 (0.03)	0.47 (0.05)	0.28 (0.05)	0.46 (0.04)
RAP	**0.47** (0.04)	**0.64** (0.06)	**0.19** (0.04)	**0.58** (0.03)
RAP (Approx)	**0.48** (0.05)	**0.63** (0.07)	**0.20** (0.04)	**0.55** (0.05)

SMBO, Sequential model-based optimization; CV, cross-validation. Bold values are for performance which is more than 1 standard deviation better than alternative methods.

Case 1. Normal linear regression. In the case of linear regression we have that $g(\cdot)$ is the identity such that $\theta_{t+1} = X_{t+1}^T \widehat{\beta}_t(\lambda_t)$ and C_{t+1} is defined as in Section 4.1.

Case 2. Logistic regression. In this case we have that $g(\cdot)$ is the logistic function. As before $\theta_{t+1} = X_{t+1}^T \widehat{\beta}_t(\lambda_t)$ and the negative log-likelihood is defined as:

$$C_{t+1} = C(X_{t+1}, y_{t+1}) = -y_{t+1} X_{t+1} \widehat{\beta}_t(\lambda_t) + \log(1 + e^{X_{t+1}^T \widehat{\beta}_t(\lambda_t)}).$$

Proposition 2. *[Adapted from ref. [25]] In the context of ℓ_1 penalized GLM models, the derivative $\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda_t}$ is also available in closed form as follows:*

$$\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda_t} = -(X_{1:t}^T W X_{1:t})^{-1} \operatorname{sign}(\widehat{\beta}_t(\lambda)) \tag{15}$$

where W is a diagonal matrix with entries $w_i V_i^{-1} \left(\frac{\partial \mu_i}{\partial \eta_i} \right)$.

Proof. The proof is closely related to that of Proposition 1. A full derivation is provided in Appendix A. □

Remark 1. We note that applying the RAP framework in the context of GLMs requires only minor modifications from the procedure detailed Algorithm 1.

5 | EMPIRICAL RESULTS

In this section we empirically demonstrate the capabilities of the proposed framework via a series of simulations. We begin by considering the performance of the RAP algorithm in the context of stationary data. This simulation serves to demonstrate that the proposed method is capable of accurately tracking the regularization parameter. We then study the performance of RAP algorithm in the context of nonstationary data. Throughout this simulation study the RAP algorithm is benchmarked against two offline methodologies: cross-validation and SMBO. In the context of SMBO methods, we study the performance against Bayesian optimization methods. Here a Gaussian process with a square exponential kernel was employed as a surrogate model together with the expected improvement acquisition function.

5.1 | Simulation settings

In order to thoroughly test the performance of the RAP algorithm, we look to generate synthetic data were we are able to control both the underlying structure as well as the dimensionality of the data. In this work, the covariates X_t were generated according to a multivariate Gaussian distribution with a block covariance structure. This introduced significant correlations across covariates, thereby increasing the difficultly of the regression task. Formally, the data simulation process followed that described by ref. [16]. This involved sampling each covariate as follows:

$$X_t \sim \mathcal{N}(0, \Sigma),$$

where $\Sigma \in \mathbb{R}^{p \times p}$ is a block diagonal matrix consisting of five equally sized blocks. Within each block, the off-diagonal entries were fixed at 0.8, while the diagonal entries were fixed to be one. Having generated covariates, X_t, a sparse vector of regression coefficients, β, was simulated. This involved randomly selecting a proportion, ρ, of coefficients and randomly generating their values according to a standard Gaussian distribution. All remaining coefficients were set to zero. Given simulated covariates, X_t, and a vector of sparse regression coefficients β, the response was simulated according to an exponential family distribution with mean parameter $\mu_t = g^{-1}(X_t \beta)$. In this manner, data was generated from both a Gaussian as well as Binomial distributions. In case of the former, we therefore have that $y_t \sim \mathcal{N}(X_t^T \beta, 1)$, while in the case of logistic regression we have that y_t follows a Bernoulli distribution with mean $\sigma(X_t^T \beta)$ where $\sigma(\cdot)$ denotes the logistic function.

In this manner, it is possible to generate piecewise stationary data, $\{(y_t, X_t): t = 1, \ldots, T\}$. When studying the performance of the RAP algorithm in the context of stationary data, it sufficed to simulate one such data set. In order to quantify performance in the context of nonstationary data, we concatenate multiple piece-wise stationary data sets, resulting in data sets with abrupt changes. We note that for each piece-wise stationary data set distinct regression coefficients, with varying sparse supports, were simulated. We note that in the nonstationary setting the block structure was randomly

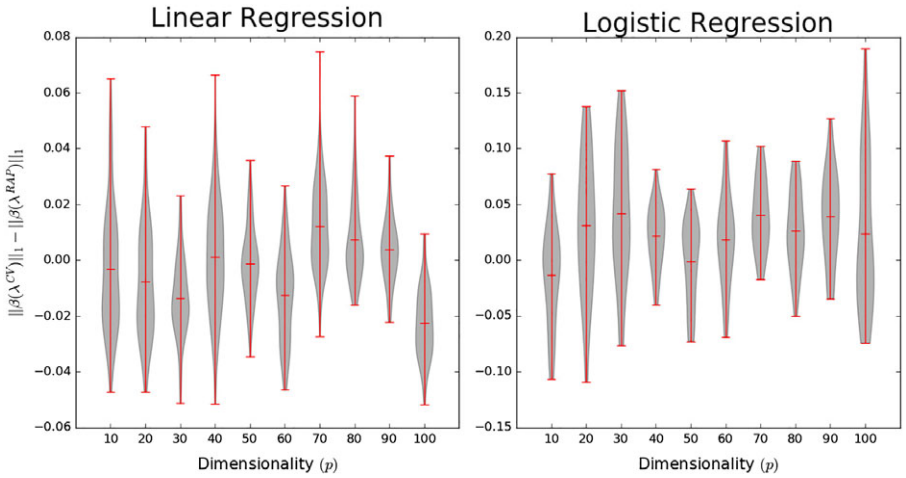

FIGURE 1 Violin plots visualizing the difference in selected regularization parameters as a function of the dimensionality, p, for linear (A) and logistic regression (B). We note that the difference in estimated ℓ_1 norms is both small and centered around the origin, indicating the absence of large systematic bias. Note the difference in y-axis across panels

permuted at each iteration to avoid covariates sharing the same set of correlated variables.

5.2 | Performance metrics

In order to assess the performance of the RAP algorithm we consider various metrics. In the context of stationary data, our primary objective is to demonstrate that the proposed method is capable of tracking the regularization parameter when benchmarked against traditional methods such as cross-validation. As a result, we consider the difference in ℓ_1 norms of the regression model estimated by each algorithm. This is defined as:

$$\Delta = \|\beta(\lambda^{CV})\|_1 - \|\beta(\lambda^{RAP})\|_1, \tag{16}$$

where we write λ^{CV} and λ^{RAP} to denote the regularization parameters selected by cross-validation and RAP algorithms, respectively. We choose to employ the ℓ_1 norm (as opposed to directly considering the sparsity parameter, λ) as there is a one-to-one relationship between λ and the ℓ_1 norm. This serves to bypass any potential issues arising from scaling or other idiosyncrasies.

In the context of nonstationary data we are interested in two additional metrics. The first corresponds to the negative log-likelihood of each new unseen observations, C_{t+1}, initially defined in Equation (6). Secondly, we also consider the correct recovery of the sparse support of β_t. In this context, we treat the recovery of the support of β_t as a binary classification problem and quantify the performance using the F score; defined as the harmonic mean between the precision and recall of a classification algorithm.

5.3 | Stationary data

We begin by demonstrating that the RAP framework is capable of accurately tracking the regularization parameter in the context of stationary data. In particular, we study the performance of the RAP algorithm as the dimensionality of regression coefficients, p, increases.

Data was generated as described in Section 5.1 and the dimensionality of the covariates, X_t, was varied from $p = 10$ through to $p = 100$. For each value of p, data sets consisting of $n = 300$ observations where randomly generated. The regularization parameter was first estimated using $K = 10$ fold cross-validation. The RAP algorithm was subsequently employed and the difference in ℓ_1 norm, defined in Equation (16), was then computed. In case the of the RAP algorithm, each observation was studied once in a streaming fashion. Due to the stationary nature of the data, it was not imperative to discard past observations in this setting. As a result, the sample covariance matrix was estimated using a fixed forgetting factor of $r = 1$ in these experiments. Given such a choice of r, the updates for the regularization parameter are equivalent to those proposed by ref. [7] in the case of linear regression. The initial choice for the regularization parameter, λ_0, was randomly sampled from a uniform distribution, $\mathcal{U}[01]$. Both normal linear and logistic regression were studied in this manner.

The difference in selected regularization parameters over $N = 500$ simulations is visualized in Figure 1. It is reassuring to note that, for both linear and logistic regression, the differences are both small in magnitude as well as centered around the origin. This serves to indicate the absence of a large systematic bias. However, we note that there is higher variance in the context of logistic regression.

5.4 | Nonstationary data

While Section 5.3 provided empirical evidence demonstrating that the RAP framework can be effectively employed to track regularization parameters in a stationary setting, we are ultimately interested in streaming, nonstationary data sets. As a result, in this simulation we study the performance of the proposed framework in the context of nonstationary data. As

in Section 5.3 we study the properties of the RAP algorithm in the context of linear and logistic regression.

While there are a multitude of methods through which to simulate nonstationary data, in this simulation study we chose to generate data with piece-wise stationary covariance structure. As a result, the underlying covariance alternated between two regimes: a sparse regime where the response was driven by a reduced subset of covariates and a dense regime where the converse was true. Thus, pairs (y_t, X_t) of response and predictors were simulated in a piece-wise stationary regimes. The dimensionality of the covariates was fixed at $p = 20$, implying that $X_t \in \mathbb{R}^{20}$. Changes occurred abruptly every 100 observations and two change points were considered, resulting in 300 observations in total.

Covariates, X_t, were simulated as described in Section 5.1 within two alternating regimes; dense and sparse. The block-covariance structure remained fix within each regime (ie, for 100 observations). Within the dense regime, a proportion $\rho_1 = .8$ of regression coefficients were randomly selected and their values sampled from a standard Gaussian distribution. All remaining coefficients were set to zero. Similarly, in the case of the sparse regime, $\rho_2 = .2$ regression coefficients were randomly selected with remaining coefficients set to zero. The regression coefficients remained fixed within each regime.

In order to benchmark the performance of the proposed RAP framework, streaming penalized Lasso models were also estimated using a fixed and stepwise constant sparsity parameters. As a result, the RAP algorithm was benchmarked against three distinct offline methods for selecting the regularization parameter. In the case of a fixed sparsity parameter, $K = 10$-fold cross-validation as well as Bayesian optimization were employed. Finally, cross-validation was also employed to learn a stepwise constant regularization parameter. This was achieved by performing cross-validation for the data within each regime. Such an approach requires knowledge of the piece-wise stationary nature of the data that would not typically be available, and we therefore refer to it as the *Stepwise Oracle* method. Moreover, the offline nature of each of these methods, dictated that the entire data set should be analyzed simultaneously (as opposed to in a streaming fashion by the RAP algorithm). As such, they serve to provide a benchmark but would be infeasible in the context of streaming data.

Further, we also study the performance of the RAP framework in three distinct settings. First, we study the use of the exact gradient update provided in Equation (10) as well as the approximate gradient update described in Equation (11). These two settings both employ a fixed forgetting factor to iteratively estimate the covariance matrix as detailed in Equation (4) where we set the fixed forgetting factor to be $r = .95$. Finally, we also consider the scenario where past information is not discarded. This corresponds to setting $r = 1$ as in Section 5.3 and is equivalent to the update described by ref. [7] in the case of linear regression. This benchmark serves to demonstrate the importance of discarding past observations.

Results for $N = 500$ simulations are shown in Figure 2. The estimated time-varying regularization parameter for both the linear and logistic regression models is shown on the left panels. These results provide evidence that the RAP algorithm is able to reliably track the piece-wise constant regularization parameters selected by the *Stepwise Oracle* using cross-validation (shown in red). We note that there is some lag directly after each change occurs, however, the estimated regression parameters are able to adapt thereafter. However, this is not the case when past information is not discarded, corresponding to the use of $r = 1$ as a forgetting factor. This is to be expected as information is averaged across several distinct regimes meaning the regression model is misspecified. Figure 2 also shows the mean negative log-likelihood over unseen samples, C_{t+1}. We note there are abrupt spikes every 100 observations, corresponding to the abrupt changes in the underlying dependence structure. Detailed results are provided in Table 1. We note that the proposed framework is able to outperform the alternative offline approaches. In the case of the offline cross-validation and SMBO, this is to be expected as a fixed choice of regularization parameter is misspecified.

6 | APPLICATION TO FMRI DATA

In this section we present an application of the RAP algorithm to task-based functional MRI (fMRI) data. This data corresponds to time series measurements of blood oxygenation, a proxy for neuronal activity, taken across a set of spatially remote brain regions. Our objective in this work is to quantify pairwise statistical dependencies across brain regions, typically referred to as functional connectivity within the neuroimaging literature [28].

While traditional analysis of functional connectivity was rooted on the assumption of stationarity, there is growing evidence to suggest this is not the case [12]. This particularly true in the context of task-based fMRI studies. Several methodologies have been proposed to address the nonstationary nature of fMRI data [21], many of which are premised on the use of penalized regression models such as those studied in this work. While such methods have made important progress in the study of nonstationary connectivity networks, they have typically employed fixed regularization parameters. This is difficult to justify in the context of nonstationary data and plausible biological justifications are not readily available. The RAP algorithm is therefore ideally suited to both accurately estimating nonstationary connectivity structure as well as providing insight regarding whether the assumption of a fixed sparsity parameter is reasonable.

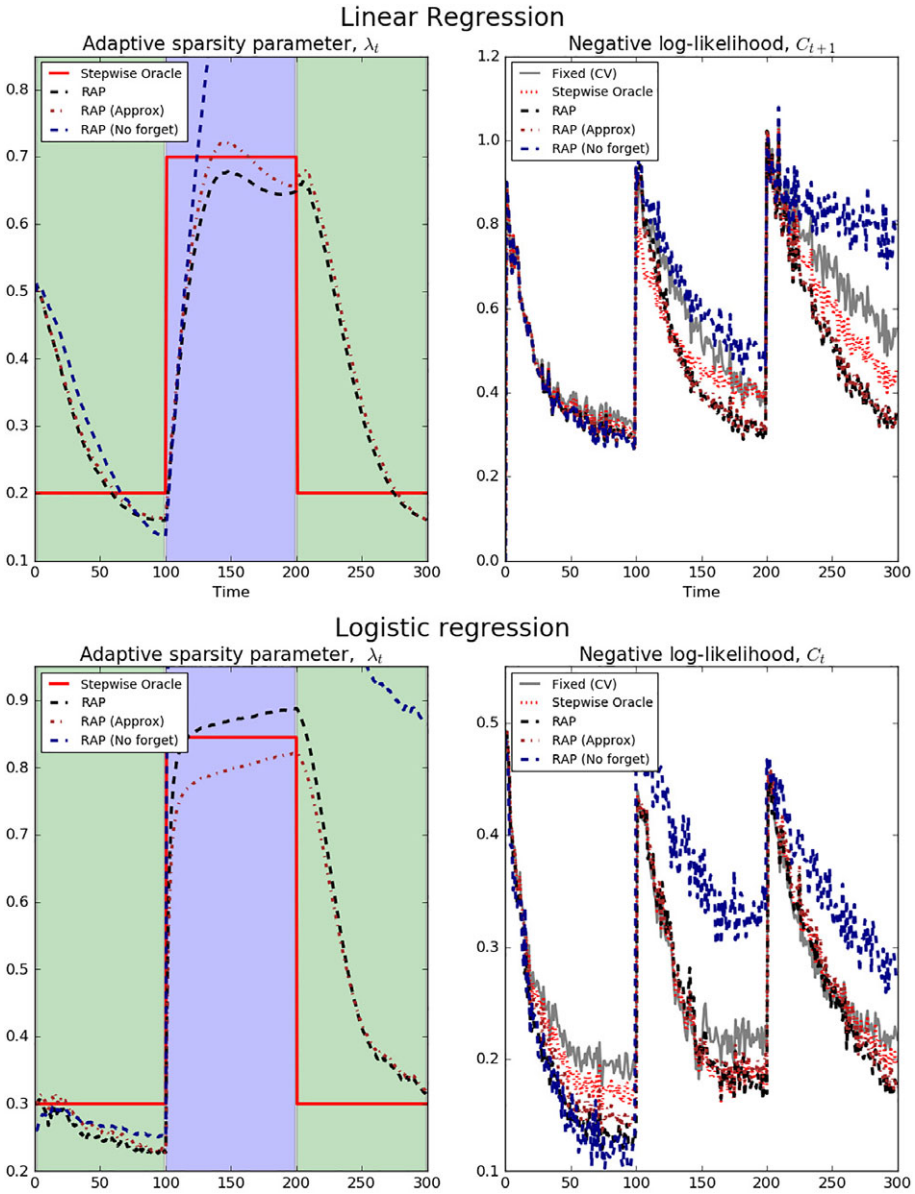

FIGURE 2 Simulation results when estimating regularized streaming linear and logistic regression models. Results for linear and logistic regression are shown across the first and second rows, respectively. The left panels plot the mean regularization parameter as estimated by the RAP algorithm as well as the optimal piece-wise constant value selected by a *Stepwise Oracle* using cross-validation. The right panels plot the mean negative log-likelihood, C_{t+1}, over time. We note that the RAP algorithms outperform the offline alternatives. We note that results for SMBO are omitted for clarity. Detailed results are provided in Table 1

6.1 | Estimating connectivity via Lasso regressions

Estimating functional connectivity networks is fundamentally a statistical challenge. A functional relationship is said to exist across two spatially remote brain regions if their corresponding time series share some statistical dependence. While this can be quantified in a variety of ways, a popular approach is the use of Lasso regression models to infer the conditional independence structure of a particular node. In such an approach, the time series of a given node is regressed against the time series of all remaining nodes. A functional relationship is subsequently inferred between the target node and all remaining nodes associated with a nonzero regression coefficient. The connectivity structure across all nodes can then be inferred via a neighborhood selection approach [17]. The proposed RAP framework can directly be incorporated into such a model, resulting in time varying conditional dependence structure where the underlying sparsity parameter is also inferred.

6.2 | Human Connectome Project Emotion Task Data

Emotion task data from the Human Connectome Project was studied with 20 subjects selected at random. During the task participants were presented with blocks of trials that either required them to decide which of two faces presented on the bottom of the screen match the face at the top of the screen, or which of two shapes presented at the bottom of the screen match the shape at the top of the screen. The faces had either an angry or fearful expression while the shapes represented the emotionally neutral condition. Twenty regions

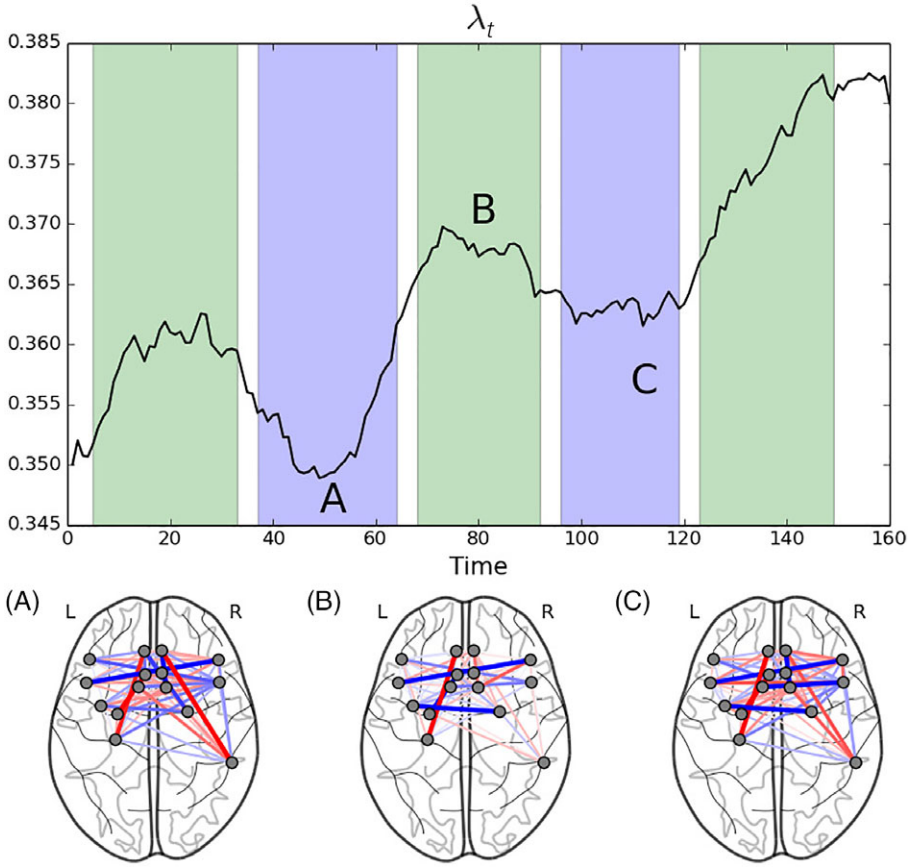

FIGURE 3 Top: the mean sparsity parameter is shown as a function of time. The background color indicates the nature of the task at hand (green indicates neutral task while blue indicates the emotion task). Bottom: estimated networks visualizing the estimated connectivity structure at three distinct points in time. Edge colors indicate the nature of the dependence (blue indicates a positive dependence, red a negative dependence)

were selected from an initial subset of 84 brain regions based on the Desikan-Killiany atlas. Data for each subject therefore consisted of $n = 175$ observations across $p = 20$ nodes.

6.3 | Results

Data for each subject was analyzed independently where the time-varying estimates of the conditional dependence structure for each node were estimated as described in Section 5.1. A fixed forgetting factor of $r = .95$ was employed throughout with a stepsize parameter $\epsilon = .025$. The exact gradient was employed when updating the sparsity parameter at each iteration.

The mean sparsity parameter over all subjects is shown in the top panel of Figure 3. We observe decreased sparsity parameters for blocks in which subjects were presented with emotional (ie, angry or fearful) faces (top panel, purple shaded areas) as compared to blocks in which subjects were shown neutral shapes (top panel, green shaded areas). The oscillation in sparsity parameter is highly correlated with task onset. When inspecting the networks estimated using the time varying sparsity parameter (bottom panel), we find strong coupling among many of the regions during the emotion processing blocks (A and C) compared to a clearly sparser network representation for blocks that require no emotion processing (ie, neutral shapes, block B). This is to be expected as the selected regions are core hubs involved with emotion processing; therefore explaining the higher network activity during the emotion task.

7 | CONCLUSION

In this work we have presented a framework through which to learn time-varying regularization parameters in the context of streaming GLMs. An approximate algorithm is also provided to address issues concerning computational efficiency. We present two simulation studies which demonstrate the capabilities of the RAP framework. These simulations show that the proposed framework is capable of tracking the regularization parameter both in a stationary as well as nonstationary setting. Finally, we present an application to task-based fMRI data, which is widely accepted to be nonstationary [12].

Future work will involve extending the RAP framework to consider alternative regularization schemes. In particular an ℓ_2 penalty could also be incorporated as the derivative, $\frac{\partial \hat{\beta}_i(\lambda)}{\partial \lambda}$, is available in closed form. Finally, the methods presented in this manuscript have been motivated by the study of fMRI data. It is often the case that such data is collected across multiple subjects, possibly in real-time [19]. Due to the

high-dimensional and nonstationary nature of fMRI data, it is imperative that share information across subjects in order to more reliably estimate the associated networks [23] and future work will seek to incorporate recent advances [20].

ORCID

Ricardo P. Monti http://orcid.org/0000-0002-7823-4961

REFERENCES

1. Charu Aggarwal, *Data streams: models and algorithms*, Vol 31, Springer Science & Business Media, New York, USA, 2007.
2. Leon Bottou, *Online learning and stochastic approximations*, Online Learn. Neural Netw. 17(9) (1998), 142.
3. Leon Bottou, *Large-scale machine learning with stochastic gradient descent*, COMPSTAT'2010, Springer, 2010, pp. 177–186.
4. John Duchi, Elad Hazan, and Yoram Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, J. Mach. Learn. Res. 12 (2011), 2121–2159.
5. Bradley Efron et al., *Least angle regression*, Ann. Stat. 32(2) (2004), 407–499.
6. Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, J. Stat. Softw. 33(1) (2010), 1–22.
7. Pierre Garrigues and Laurent El Ghaoui, *An homotopy algorithm for the lasso with online observations*, Advances in Neural Information Processing Systems, 2009, pp. 489–496.
8. Wolfgang Härdle, Weining Wang, and Lining Yu, *Tenet: Tail-event driven network risk*, J. Econom. 192(2) (2016), 499–513.
9. Trevor Hastie, Robert Tibshirani, and Martin Wainwright, *Statistical learning with sparsity: The lasso and generalizations*, CRC Press, Boca Raton, Florida, USA, 2015.
10. Simon Haykin, *Adaptive filter theory*, Pearson, India, 2008.
11. Nicholas Heard et al., *Bayesian anomaly detection methods for social networks*, Ann. Appl. Stat. 4(2) (2010), 645–662.
12. Matthew Hutchison et al., *Dynamic functional connectivity: promise, issues, and interpretations*, NeuroImage 80 (2013), 360–378.
13. Romy Lorenz et al., *The automatic neuroscientist: A framework for optimizing experimental design with closed-loop real-time fmri*, NeuroImage 129 (2016), 320–334.
14. Romy Lorenz et al., *Dissociating frontoparietal brain networks with neuroadaptive bayesian optimization*, Nature Commun. 9(1) (2018), 1227.
15. Peter. McCullagh and John Nelder, *Generalized linear models*, Vol 37, CRC Press, London, UK, 1989.
16. Brian McWilliams, Christina Heinze, Nicolai Meinshausen, Gabriel Krummenacher, and Hastagiri Vanchinathan, LOCO: Distributing ridge regression with random projections, *arXiv preprint arXiv:1406.3469*, 2014.
17. Nicolai Meinshausen and Peter Bühlmann, *High-dimensional graphs and variable selection with the lasso*, Ann. Stat. 34 (2006), 1436–1462.
18. Nicolai Meinshausen and Peter Bühlmann, *Stability selection*, J. R. Stat. Soc. Ser. B 72(4) (2010), 417–473.
19. Reed Montague et al., *Hyperscanning: Simultaneous fMRI during linked social interactions*, NeuroImage 16(4) (2002), 1159–1164.
20. Ricardo Pio Monti and Aapo Hyvärinen, A unified probabilistic model for learning latent factors and their connectivities from high-dimensional data. *arXiv preprint arXiv:1805.09567*, 2018.
21. Ricardo Pio Monti et al., *Estimating time-varying brain connectivity networks from functional MRI time series*, NeuroImage 103 (2014), 427–443.
22. Ricardo Pio Monti et al., *Real-time estimation of dynamic functional connectivity networks*, Hum. Brain Mapp. 38(1) (2016), 202–220.
23. Ricardo Pio Monti, Christoforos Anagnostopoulos, and Giovanni Montana, *Learning population and subject-specific brain connectivity networks via mixed neighborhood selection*, Ann. Appl. Stat. 11(4) (2017a), 2142–2164.
24. Ricardo Pio Monti et al., *Decoding time-varying functional connectivity networks via linear graph embedding methods*, Front. Comput. Neurosci. 11(14) (2017b), 1–12.
25. Mee Young Park and Trevor Hastie, *L1-regularization path algorithm for generalized linear models*, J. R. Stat. Soc. Ser. B Stat. Methodol. 69(4) (2007), 659–677.
26. Saharon Rosset and Zhu Ji, *Piecewise linear regularized solution paths*, Ann. Stat. 35 (2007), 1012–1030.
27. Bobak Shahriari et al., *Taking the human out of the loop: A review of bayesian optimization*, IEEE Proc. 104(1) (2016), 148–175.
28. Steven Smith et al., *Network modelling methods for fMRI*, NeuroImage 54(2) (2011), 875–891.
29. Robert Tibshirani, *Regression shrinkage and selection via the lasso*, J. R. Stat. Soc. Ser. B (1996), 58, 267–288.
30. Nikolaus Weiskopf, *Real-time fMRI and its application to neurofeedback*, NeuroImage 62(2) (2012), 682, 1–692.

How to cite this article: 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;1–11. https://doi.org/10.1002/sam.11390

APPENDIX A: PROOF OF PROPOSITION 2

For a given regularization parameter, λ, the corresponding vector of estimated regression coefficients, $\widehat{\beta}(\lambda)$, can be computed by minimizing the nonsmooth objective, $L_t(\beta, \lambda)$, provided in Equation (14). The subgradient is defined as:

$$\nabla_\beta L_t(\beta, \lambda) = -X_{1:t}^T W(y_{1:t} - \boldsymbol{\mu})\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\mu}} + \lambda \ \text{sign}(\beta) \tag{A1}$$

We write $\boldsymbol{\mu}$ to denote the vector of predicted means, $\mu_i = g^{-1}(\eta_i) = g^{-1}(X_i^T\beta)$ and $\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\mu}}$ to denote a vector with entries $\frac{\partial \boldsymbol{\eta}_i}{\partial \boldsymbol{\mu}_i}$. Note that in the case of normal linear regression we have that $\mu_i = \eta_i = X_i^T\beta$ and we therefore recover Equation (9).

As in Proposition 1, we have that for any choice of regularization parameter, the subgradient evaluated at $\widehat{\beta}_t(\lambda)$ must satisfy:

$$\nabla_\beta L_t(\beta, \lambda)|_{\beta=\widehat{\beta}_t(\lambda)} \ni 0.$$

We therefore compute the derivative with respect to λ in order to obtain:

$$\frac{\partial}{\partial \lambda}(\nabla_\beta L_t(\beta, \lambda)|_{\beta=\widehat{\beta}_t(\lambda)}) = 0 \tag{A2}$$

$$= \frac{\partial}{\partial \lambda}\left(-X_{1:t}^T W(y_{1:t} - \boldsymbol{\mu})\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\mu}}\right) + \text{sign}(\widehat{\beta}_t(\lambda)) \tag{A3}$$

$$= X_{1:t}^T W \frac{\partial \boldsymbol{\mu}}{\partial \lambda}\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\mu}} + \text{sign}(\widehat{\beta}_t(\lambda)) \tag{A4}$$

$$= X_{1:t}^T W X_{1:t}\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda} + \text{sign}(\widehat{\beta}_t(\lambda)) \tag{A5}$$

where Equation (A5) follows from the fact that:

$$\frac{\partial \boldsymbol{\mu}}{\partial \lambda} = \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}\frac{\partial \boldsymbol{\eta}}{\partial \widehat{\beta}_t(\lambda)}\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda} = \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}} X_{1:t}\frac{\partial \widehat{\beta}_t(\lambda)}{\partial \lambda}.$$

Rearranging Equation (A5) yields the result.