Online-Forgetting Process for debiased-Lasso using Summary Statistics

Xiao Guo, Xu Zhang and Hai Zhang*

School of Mathematics, Northwest University, China

Abstract: Data removal and data forgetting have become standard requests for users, since the enactment of regulations such as GDPR and CCPA. In this paper, we study the data removal problem for statistical inference of lasso within an *online-forgetting* framework, where the new data batch arrives sequentially while the earliest data batch is removed to ensure a constant memory constraint. We propose a new algorithm, dOnFL, which has several appealing properties: it is computationally efficient compared to retraining the model, and it avoids accessing the full data batches by utilizing only the current batch and the summary statistics of historical batches within the available time frame. In particular, we develop an efficient debiasing technique to reduce the bias induced by the ℓ_1 penalty of lasso. Theoretically, we establish the asymptotic normality of the proposed estimator as the total sample size of available data batches goes to infinity. The simulation and real data experiments demonstrate the merits of the proposed algorithm.

*Corresponding author (zhanghai@nwu.edu.cn).

Key words and phrases: Data removal, Summary statistics, Statistical inference, Lasso

1. Introduction

Advancements in computing and measurement technologies have led to a surge in streaming data collected over time across various fields, such as the user data from internet companies and organizations. On the other hand, legal requirements, including the EU's General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), mandate that companies and organizations should remove and forget certain user personal data to protect their privacy. Specifically, users have the right to request that the platform retain their personal information only for a limited time. Therefore, statistical and machine learning methods should be developed in an *online-forgetting* fashion.

In recent years, there has been a growing interest in developing methods that address data forgetting and data removal requirements. Specifically, Li et al. (2021) proposed an online-forgetting process for linear regression models, where they considered the First-in-First-Delete (FIFD) scenario, namely, the data owner is required to delete the oldest data batch upon

https://eur-lex.europa.eu/eli/reg/2016/679/oj
https://oag.ca.gov/privacy/ccpa

receiving the newest data batch to maintain a fixed memory limit. The proposed method achieves *perfect* forgetting by yielding the same solution as retraining the model using the available data. Guo et al. (2020) studied the problem in a framework called certified removal, where the model, after data removal, is indistinguishable from a model that never seen the data to begin with. They considered the L_2-regularized linear and logistic models and developed a Newton update on the model parameters to remove the influence of the deleted data point. There exists several other work on data removal; see Ginart et al. (2019); Liu and Tsaftaris (2020); Izzo et al. (2021); Bourtoule et al. (2021), among others.

Most of the aforementioned work focused on the parameter estimation in statistical or machine learning models using low-dimensional data. However, high-dimensional data are increasingly prevalent across various application fields. In addition, apart from the parameter estimation, the statistical inference is also vital and useful to quantify the uncertainty of the estimated parameters.

Motivated by this, we consider the statistical inference in high-dimensional regression under the FIFD scenario (Li et al., 2021). To solve this problem, the online-forgetting statistical inference method should possess the following three desirable properties. The first is computational efficiency.

A naive way to achieve *perfect* forgetting is to retrain the model. However, when the data batches sequentially arrive, retraining the model becomes extremely time-consuming. Therefore, the method should be time-economic. The second is statistical accuracy. The statistical accuracy is expected to be sacrificed in order to pursue the computational efficiency. Thus, the goal is to seek a trade-off between the computational efficiency and the statistical accuracy. In particular, to facilitate the statistical inference, the estimator should be unbiased asymptotically. The third is ~~summary statistics compatibility~~. Due to privacy and data avaliablity issues, the method should be based on the summary statistics rather than the raw data.

The contributions of this work can be summarized as follows. First, we develop a new algorithm called **D**ebiased **On**line-**F**orgetting **L**asso (dOnFL), tailored for statistical inference under the scenario where the data batch sequentially arrives and the earliest data batch are required to be removed upon the arrival of new data batch. The algorithm only requires the current data batch and the summary statistics of all historical data batches within the available time frame, thereby mitigating concerns related to data availability and privacy. By eliminating the summary statistics corresponding to the data batches to be removed, the algorithm achieves the data forgetting requirement. To the best of our knowledge, this is the first algorithm for the data removal in high-dimensional regression models. It is worth mentioning that the online algorithms for lasso have been studied by Langford et al. (2009); Duchi et al. (2011); Tarres and Yao (2014); Sun et al. (2024), among others. In these algorithms, the parameters are updated using new data and the most recently updated parameters, which rely on old data and therefore do not satisfy the data forgetting requirements.

Second, we develop a new and efficient debiasing technique to reduce the bias induced by the ℓ_1 penalty of lasso, in order to facilitate the following-up statistical inference. In the offline setting, various bias-correction methods have been proposed; see Javanmard and Montanari (2014); van de Geer et al. (2014); Zhang and Zhang (2014), among others. However, most methods require the availability of the entire dataset. Even without this requirement, applying these offline techniques to the current data batch and available summary statistics would be rather time consuming in the online-forgetting scenario, where the estimator should be updated in time once the newest data batch arrives and the oldest data batch is removed. In the online setting, various debiasing techniques have also been proposed; see van de Geer et al. (2014); Chen et al. (2020); Shi et al. (2021); Deshpande et al. (2023); Han et al. (2024), among others. However, similar to the offline debiasing techniques, most online debiasing algorithms require the entire data and the online algorithms implicitly use all the previous data. To alleviate these issues, when the new data batch arrives, we use the new data batch combined with the previously computed summary statistics of the remaining available data batches to debias.

Last but not least, we provide the asymptotic normality of the proposed estimator as the total sample size of available data batches goes to infinity, under certain conditions. Based on the asymptotic normality, we construct the confidence interval of the proposed estimator, in order to quantify its uncertainty. The effectiveness of the constructed confidence interval is validated by experiments.

The remainder of this paper is organized as follows. Section 1.1 introduces the notations. Section 2 presents the online-forgetting process for lasso. Section 3 establishes the asymptotic property of the proposed method. Sections 4 and 5 evaluates the proposed method via simulation and real data experiments. Section 6 concludes the paper. All the proofs, technical lemmas, and additional simulations are provided in the Supplementary Materials.

1.1 Notations

The following notations are generally needed. $\|A\|_2$ denotes the spectral norm of the matrix A or the Euclidian norm of vector A, $\|A\|_\infty$ denotes the infinity norm of the matrix A, and $\|A\|_0$ denotes the number of non-zero entries of the matrix or vector A. We write $f(n) \asymp g(n)$ or $f(n) = \Theta(g(n))$ if $cg(n) \leq f(n) \leq Cg(n)$ for some constants $0 < c < C < \infty$; $f(n) \lesssim g(n)$ or $f(n) = O(g(n))$ if $f(n) \leq Cg(n)$ for some constant $C < \infty$; and $f(n) \gtrsim g(n)$ or $f(n) = \Omega(g(n))$ if $f(n) \geq cg(n)$ for some constant $c > 0$. Moreover, $f(n) = o(g(n))$ if $f(n)/g(n) \to 0$ as $n \to \infty$; $f(n) = o_p(g(n))$ if $f(n)/g(n) \to 0$ with probability approaching to one as $n \to \infty$.

2. Online-Forgetting Process for lasso

In this section, we develop the online-forgetting process for the lasso problem. In particular, the online-forgetting process for parameter estimation and bias-correction are proposed, respectively. Afterwards, the confidence interval for the debiased estimator is obtained.

2.1 Parameter Estimation

Suppose the data batches arrive sequentially. At time stamp $1 \leq b < \infty$, the data batch $D_b = \{y^{(b)}, X^{(b)}\}$ is arrived. Considering the following online-forgetting scheme. At time stamp b, the data batches with time stamps more than T prior to b are forgotten. That is, the available data batches are $D_{[a,b]} = \{D_a, ..., D_b\}$, where $a = b+1-T$ if $b \geq T$ and $a = 1$ if $b < T$. In addition, due to the data avaliability and privacy issue, for time stamps in the range $[a, b-1]$, only the summary statistics are available. For all $1 \leq b < \infty$, the samples in data batch b are independently generated from the following high-dimensional linear model,

$$y^{(b)} = X^{(b)}\beta_0 + \epsilon^{(b)},$$

where $y^{(b)} \in \mathbb{R}^{n_b}$ is a response vector, $X^{(b)} = (x_1^{(b)}, \ldots, x_p^{(b)}) \in \mathbb{R}^{n_b \times p}$ including n_b samples of dimension p, the true parameter $\beta_0 \in \mathbb{R}^p$ is sparse and unknown, and the error term $\epsilon^{(b)}$ has *i.i.d.* entries with mean 0 and variance σ^2.

If the original data batches $D_{[a,b]}$ are all available, we can in principle use the following offline lasso to obtain an estimator $\hat{\beta}^{[a,b]}$ of β_0,

$$\hat{\beta}^{[a,b]}(\lambda_{[a,b]}) = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2\sum_{j=a}^{b} n_j} \sum_{j=a}^{b} \|y^{(j)} - X^{(j)}\beta\|_2^2 + \lambda_{[a,b]}\|\beta\|_1 \right\}, \tag{2.1}$$

where we use the subscript $[a, b]$ to emphasis that the tuning parameter λ

depends on the data batches $D_{[a,b]}$. However, in our set-up, the original data except the data except the data batch D_b are not available. Instead, we only have access to the summary statistics. Define

$$\mathcal{SS}^{[a,b]} := \left\{S^{[a,b]}, U^{[a,b]}\right\}$$
$$\text{with} \quad S^{[a,b]} := \sum_{j=a}^{b} (X^{(j)})^T X^{(j)} \text{ and } U^{[a,b]} := \sum_{j=a}^{b} (X^{(j)})^T y^{(j)}. \tag{2.2}$$

With the summary statistics $\mathcal{SS}^{[a,b]}$, we can then obtain $\hat{\beta}_r^{[a,b]}$ by the Iterative Shrinkage-Thresholding Algorithm (ISTA) (see, e.g., Beck and Teboulle (2009); Parikh et al. (2014)), i.e., repeating the following two steps:

- Step 1: $\hat{\beta}^{[a,b]} \leftarrow \hat{\beta}^{[a,b]} - \frac{\eta_b}{\sum_{j=a}^{b} n_j}(S^{[a,b]}\hat{\beta}^{[a,b]} - U^{[a,b]})$,
- Step 2: $\hat{\beta}_r^{[a,b]} = \text{Soft}(\hat{\beta}_r^{[a,b]}, \eta_b \lambda_{[a,b]}) := \text{sign}(\hat{\beta}_r^{[a,b]}) \max\{0, |\hat{\beta}_r^{[a,b]}| - \eta_b \lambda_{[a,b]}\}$,

where $\hat{\beta}_r^{[\mathrm{a,b}]} (r = 1, \ldots, p)$ is the r-th component of $\hat{\beta}^{[\mathrm{a,b}]}$ and η_b is the step size. Note that to ensure the data forgetting, the initial value of $\hat{\beta}^{[a,b]}$ is set to zero at each time step, which effectively eliminates the possibility that parameters retain information from the previously observed but forgotten data. These two steps are carried out iteratively until convergence.

Remark 1. *The ISTA algorithm belongs to the more general proximal gra-*

dient descent algorithm (Parikh et al., 2014). For the optimization problem

$$\min_x f(x) + g(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are closed proper convex and f is differentiable, it has been shown that the proximal gradient algorithm with fixed step size converges provided that the step size $\eta \in (0, L]$ with L being the Lipschitz constant of $\nabla f(x)$ (Parikh et al., 2014). Following this theoretical guidance, in practice, we fix η_b in the ISTA algorithm as

$$\eta_b = \frac{1}{\lambda_{\max}((X^{[a,b]})^T X^{[a,b]})}, \tag{2.3}$$

where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of the corresponding matrix.

Remark 2. *In paractice, the tuning parameter $\lambda_{[a,b]}$ in (2.1) and the ISTA algorithm can be selected by minimizing the prediction error of data batch D_b using the estimator sequence $\hat{\beta}^{[a,b-1]}(\lambda)$ obtained from previous data batches $D_{[a,b-1]}$, i.e.,*

$$\arg\min_{\lambda \in \tilde{\lambda}} \frac{1}{n_b} \|y^{(b)} - X^{(b)} \hat{\beta}^{[a,b-1]}(\lambda)\|_2^2, \tag{2.4}$$

where $\tilde{\lambda}$ denotes the set of candidate values.

From Steps 1 and 2, it is evident that the parameter estimation entirely depends on the summary statistics. When the new data batch D_{b+1} ($b \geq T$) arrives and the most previous data batch D_a is forgotten, namely, the available data bathes are $D_{[a+1,b+1]} = \{D_{a+1}, \ldots, D_b, D_{b+1}\}$, we only need to update the summary statistics as follows:

$$S^{[a+1,b+1]} := S^{[a,b]} + S^{(b+1)} - S^{(a)} \quad \text{and} \quad U^{[a+1,b+1]} := U^{[a,b]} + U^{(b+1)} - U^{(a)}, \tag{2.5}$$

where $S^{(j)}$ and $U^{(j)}$ are defined by (2.2) with $a = b = j$. By substituting the updated summary statistics $S^{[a+1,b+1]}$ and $U^{[a+1,b+1]}$ into Step 1 and Step 2, the new estimator $\hat{\beta}_r^{[a+1,b+1]}(r = 1, \ldots, p)$ can be obtained.

2.2 Bias-correction

It is well-known that the lasso estimator is biased, which is harmful for statistical inference problems such as the interval estimation. Therefore, we need to correct the bias of the estimator $\hat{\beta}^{[a,b]}$ obtained in Section 2.1.

In the offline setting, Zhang and Zhang (2014) proposed a method called LDPE for removing the bias of lasso. However, LDPE is not suitable for the online-forgetting set-up considered in this work. First, LDPE used all the original data for the parameter estimation and bias-correction, however, the raw data is not available in our set-up. Second, upon the arrival (resp. removal) of the new data batch D_{b+1} (resp. D_a), the new training data batches become $D_{[a+1,b+1]}$. Their method corrects the bias by solving p $(p-1)$-dimensional lasso using all the raw data $D_{[a+1,b+1]}$, which is time consuming when the training data sequentially updates and the number of samples in $D_{[a+1,b+1]}$ is large.

To alleviate these issues, we develop an efficient online-forgetting process for the bias-correction based on summary statistics. At high level, to improve the computational efficiency, upon the arrival of new data batch, we use the new data batch combined with the previously computed parameters corresponding to the remaining available data batches to construct the bias-correction term. The proposed debiasing approach proceeds as follows.

Suppose the training data batches are $D_{[a,b]}$. For each data batch $D_j = \{y^{(j)}, X^{(j)}\}$, recall $x_r^{(j)}$ be the r-th column of $X^{(j)}$ and let $X_{-r}^{(j)}$ be the sub-matrix of $X^{(j)}$ excluding the r-th column. Upon the arrival of data batch D_j, conduct the following lasso regression of $x_r^{(j)}$ on $X_{-r}^{(j)}$ for each $r = 1, \ldots, p$

$$\hat{\gamma}_r^{(j)} = \underset{\gamma \in \mathbb{R}^{p-1}}{\arg\min} \left\{ \frac{1}{2n_j} \| x_r^{(j)} - X_{-r}^{(j)} \gamma \|_2^2 + \lambda_r^{(j)} \|\gamma\|_1 \right\}, \tag{2.6}$$

and denote the corresponding residual

$$\hat{z}_r^{(j)} := x_r^{(j)} - X_{-r}^{(j)}\hat{\gamma}_r^{(j)}, \tag{2.7}$$

where (2.6) can be solved efficiently by the coordinate descent algorithm (CD) (Friedman et al., 2007), among others; the tuning parameter $\lambda_r^{(j)}$ in (2.6) can be selected using data-dependent procedures applied to data batch D_j, such as AIC, BIC, or cross-validation, among others.

Our debiased estimator $\hat{\beta}_{de}^{[a,b]} = \{\hat{\beta}_{de,r}^{[a,b]}\}$ of $\hat{\beta}^{[a,b]}$ is then defined as

$$\hat{\beta}_{de,r}^{[a,b]} = \hat{\beta}_r^{[a,b]} + \left\{\sum_{j=a}^{b}(\hat{z}_r^{(j)})^T x_r^{(j)}\right\}^{-1}\left\{\sum_{j=a}^{b}(\hat{z}_r^{(j)})^T y^{(j)} - \sum_{j=a}^{b}(\hat{z}_r^{(j)})^T X^{(j)}\hat{\beta}^{[a,b]}\right\}. \tag{2.8}$$

The debiased estimator $\hat{\beta}_{de}^{[a,b]}$ is motivated by the method LDPE of Zhang and Zhang (2014). To improve the computational efficiency, we compute the residual $\hat{z}^{(j)}$ for each data batch D_j, and use the summation term $\sum_{j=a}^{b}(\hat{z}_r^{(j)})^T x_r^{(j)}$ to approximate the corresponding term of LDPE that computes the residual on all available datasets.

In the sequel, we provide the updating formulas when the new data batch D_{b+1} arrives and the oldest data batch D_a is forgotten. To that end, denote

$$\mathcal{DS}_r^{(j)} := \left\{a_{1,r}^{(j)}, a_{2,r}^{(j)}, A_r^{(j)}\right\} := \left\{(\hat{z}_r^{(j)})^T x_r^{(j)}, (\hat{z}_r^{(j)})^T y^{(j)}, (\hat{z}_r^{(j)})^T X^{(j)}\right\}. \tag{2.9}$$

The debiased estimator can then be written as

$$\hat{\beta}_{de,r}^{[a,b]} = \hat{\beta}_r^{[a,b]} + \left\{a_{1,r}^{[a,b]}\right\}^{-1}\left\{a_{2,r}^{[a,b]} - A_r^{[a,b]}\hat{\beta}^{[a,b]}\right\}, \tag{2.10}$$

which is $\hat{\beta}_r^{[a,b]}$ plus a debiasing term computed by $\mathcal{DS}_r^{[a,b]} = \left\{a_{1,r}^{[a,b]}, a_{2,r}^{[a,b]}, A_r^{[a,b]}\right\}$ with

$$\begin{aligned} a_{1,r}^{[a,b]} &:= \sum_{j=a}^{b}(\hat{z}_r^{(j)})^T x_r^{(j)} = \sum_{j=a}^{b} a_{1,r}^{(j)}, \\ a_{2,r}^{[a,b]} &:= \sum_{j=a}^{b}(\hat{z}_r^{(j)})^T y^{(j)} = \sum_{j=a}^{b} a_{2,r}^{(j)}, \\ A_r^{[a,b]} &:= \sum_{j=a}^{b}(\hat{z}_r^{(j)})^T X^{(j)} = \sum_{j=a}^{b} A_r^{(j)}. \end{aligned} \tag{2.11}$$

Therefore, we can obtain the updated debiasing statistics

$$\mathcal{DS}_r^{[a+1,b+1]} := \left\{a_{1,r}^{[a+1,b+1]}, a_{2,r}^{[a+1,b+1]}, A_r^{[a+1,b+1]}\right\}$$

via

$$a_{i,r}^{[a+1,b+1]} := a_{i,r}^{[a,b]} + a_{i,r}^{(b+1)} - a_{i,r}^{(a)}, \quad A_r^{[a+1,b+1]} := A_r^{[a,b]} + A_r^{(b+1)} - A_r^{(a)}, \quad i = 1, 2. \tag{2.12}$$

The whole procedure is summarized in Algorithm 1. We now compare the time complexity of dOnFL with its offline counterpart, called dLASSO algorithm. The dLASSO shares the same ISTA algorithm for parameter estimation in Stage I as dOnFL, but utilizes all available batches to compute the bias term in Stage II. See Algorithm S.1 in the Supplementary Materials for the details of the dLASSO. We consider the set-up where both of dOnFL and dLASSO update from time stamp b with available data batches being $D_{[a,b]} = \{D_a, ..., D_b\}$ to time stamp $b+1$ with available data batches being $D_{[a+1,b+1]}$, where $b-a+1 = T$. For simplicity, let the sample size $n_j = n$ for each data batch, and suppose $n < p$. For both algorithm, denote k as the number of iterates in the ISTA algorithm involved in Stage I (parameter estimation) and l as the number of iterates in the CD algorithm involved in Stage II (bias-correction). The time complexity of dOnFL turns out to be $O(lnp^2 + kp^2)$, which is lower than the time complexity $O(Tlnp^2 + kp^2)$ of dLASSO. The computational advantage of dOnFL over dLASSO mainly comes from the bias-correction stage. Details of the time complexity computation can be found in Table S.1 of the Supplementary Materials.

2.3 Interval Estimation

The effect of bias-correction is theoretically validated in Theorem 1 in Section 3, where the asymptotic normality of the debiased estimator $\hat\beta_{de}^{[a,b]}$ is established. Before introducing Theorem 1, we here would like to use the results to construct valid confidence interval for the true parameter β_0.

Based on Theorem 1, the asymptotic standard error of $\hat\beta_{de,r}^{[a,b]}$ is $\sigma\hat\tau_r^{[a,b]}$, where σ is the true standard error of the error term and

$$\hat\tau_r^{[a,b]} = \sqrt{m_r^{[a,b]}}/a_{1,r}^{[a,b]} \;\text{ with }\; m_r^{[a,b]} := \sum_{j=a}^{b} m_r^{(j)} := \sum_{j=a}^{b} (\hat z_r^{(j)})^T \hat z_r^{(j)}, \qquad (2.13)$$

where recall the definition of $\hat z_r^{(j)}$'s and $a_{1,r}^{[a,b]}$ in (2.7) and (2.11), respectively. We estimate σ using the standard estimator $\hat\sigma_\epsilon^{[a,b]}$ given as follows,

$$\begin{aligned}\hat\sigma_\epsilon^{[a,b]} &:= \left\{\frac{(y^{[a,b]} - X^{[a,b]}\hat\beta^{[a,b]})^T(y^{[a,b]} - X^{[a,b]}\hat\beta^{[a,b]})}{\sum_{j=a}^b n_j - \|\hat\beta^{[a,b]}\|_0}\right\}^{1/2}\\ &= \left\{\frac{(y^{[a,b]})^T y^{[a,b]} - 2(\hat\beta^{[a,b]})^T(X^{[a,b]})^T y^{[a,b]} + (\hat\beta^{[a,b]})^T(X^{[a,b]})^T X^{[a,b]}\hat\beta^{[a,b]}}{\sum_{j=a}^b n_j - \|\hat\beta^{[a,b]}\|_0}\right\}^{1/2}\\ &= \left\{\frac{V^{[a,b]} - 2(\hat\beta^{[a,b]})^T U^{[a,b]} + (\hat\beta^{[a,b]})^T S^{[a,b]}\hat\beta^{[a,b]}}{\sum_{j=a}^b n_j - \|\hat\beta^{[a,b]}\|_0}\right\}^{1/2}\end{aligned}$$

where $V^{[a,b]} := (y^{[a,b]})^T y^{[a,b]} = \sum_{j=a}^b (y^{(j)})^T y^{(j)} := \sum_{j=a}^b V^{(j)}$, and $U^{[a,b]}$ and

Algorithm 1 Debiased **Onl**ine-**F**orgetting **L**asso (dOnFL)

```
1:  Input: step size $\eta_{b+1}$, regularization parameters $\lambda_{[a+1,b+1]}$ and $\lambda_r^{(b+1)}$, $\mathcal{SS}^{[a,b]}$,
    $\mathcal{DS}_r^{[a,b]}$, $\mathcal{SS}^{(j)}$, $\mathcal{DS}_r^{(j)}$ and $n_j$, for $j = a, \ldots, b$ and $r = 1, \ldots, p$.
2:  Collect data batch: $D_{b+1} = \{X^{(b+1)}, y^{(b+1)}\}$.
3:  Stage I: Parameter estimation.
4:  Compute statistics $\mathcal{SS}^{(b+1)} = \left\{S^{(b+1)}, U^{(b+1)}\right\}$ with $S^{(b+1)} =$
    $(X^{(b+1)})^T X^{(b+1)}$ and $U^{(b+1)} = (X^{(b+1)})^T y^{(b+1)}$;
5:  Update statistics $\mathcal{SS}^{[a+1,b+1]} = \left\{S^{[a+1,b+1]}, U^{[a+1,b+1]}\right\}$ defined in (2.5)
    using $\mathcal{SS}^{[a,b]}$, $\mathcal{SS}^{(a)}$ and $\mathcal{SS}^{(b+1)}$;
6:  repeat
7:      Step 1: $\hat{\beta}^{[a+1,b+1]} \leftarrow \hat{\beta}^{[a+1,b+1]} - \frac{\eta_{b+1}}{\sum_{j=a+1}^{b+1} n_j}(S^{[a+1,b+1]}\hat{\beta}^{[a+1,b+1]} - U^{[a+1,b+1]})$;
8:      Step 2: $\hat{\beta}_r^{[a+1,b+1]} \leftarrow \text{Soft}\left(\hat{\beta}_r^{[a+1,b+1]}, \eta_{b+1}\lambda_{[a+1,b+1]}\right)$ for $r = 1, \ldots, p$;
9:  until convergence.
10: Stage II: Bias-correction.
11: for $r = 1$ to $p$ do
12:     Step 1: Compute $\hat{z}_r^{(b+1)} = x_r^{(b+1)} - X_{-r}^{(b+1)}\hat{\gamma}_r^{(b+1)}$, where
            $\hat{\gamma}_r^{(b+1)} = \arg\min_{\gamma \in \mathbb{R}^{p-1}} \left\{ \frac{1}{2n_{b+1}} \|x_r^{(b+1)} - X_{-r}^{(b+1)}\gamma\|_2^2 + \lambda_r^{(b+1)}\|\gamma\|_1 \right\}$;
13:     Step 2: Compute $\mathcal{DS}_r^{(b+1)} = \left\{a_{1,r}^{(b+1)}, a_{2,r}^{(b+1)}, A_r^{(b+1)}\right\}$ with
            $a_{1,r}^{(b+1)} = (\hat{z}_r^{(b+1)})^T x_r^{(b+1)}$; $a_{2,r}^{(b+1)} = (\hat{z}_r^{(b+1)})^T y^{(b+1)}$; $A_r^{(b+1)} = (\hat{z}_r^{(b+1)})^T X^{(b+1)}$
        and update $\mathcal{DS}_r^{[a+1,b+1]} = \left\{a_{1,r}^{[a+1,b+1]}, a_{2,r}^{[a+1,b+1]}, A_r^{[a+1,b+1]}\right\}$ defined in
        (2.12) using $\mathcal{DS}_r^{[a,b]}$, $\mathcal{DS}_r^{(a)}$ and $\mathcal{DS}_r^{(b+1)}$;
14:     Step 3: Using $\hat{\beta}^{[a+1,b+1]}$ to construct the debiased estimator
            $\hat{\beta}_{de,r}^{[a+1,b+1]} = \hat{\beta}_r^{[a+1,b+1]} + \left\{a_{1,r}^{[a+1,b+1]}\right\}^{-1}\left\{a_{2,r}^{[a+1,b+1]} - A_r^{[a+1,b+1]}\hat{\beta}^{[a+1,b+1]}\right\}$.
15: end for
16: Store and Clear: Store $\mathcal{SS}^{[a+1,b+1]}$, $\mathcal{DS}_r^{[a+1,b+1]}$, $\mathcal{SS}^{(j)}$, $\mathcal{DS}_r^{(j)}$, $n_j$ for $j =$
    $a+1, \ldots, b+1$ and $r = 1, \ldots, p$; and clear others.
17: Output: The non-debiased estimator $\hat{\beta}^{[a+1,b+1]}$ and debiased estimator
    $\hat{\beta}_{de}^{[a+1,b+1]}$.
```

$S^{[a,b]}$ are the summary statistics defined in (2.2). The estimator $\hat{\beta}^{[a,b]}$ is defined in Section 2.1.

Upon the arrival of the new data batch D_{b+1} and the removal of the old data batch D_a, we obtain $\hat{\beta}^{[a+1,b+1]}_{de,r}$. The estimated standard error of $\hat{\beta}^{[a+1,b+1]}_{de,r}$, denoted by $\hat{\sigma}^{[a+1,b+1]}_{\epsilon}\hat{\tau}^{[a+1,b+1]}_{r}$, can be updated using statistics as follows,

$$\hat{\sigma}^{[a+1,b+1]}_{\epsilon} = \left\{ \frac{V^{[a+1,b+1]} - 2(\hat{\beta}^{[a+1,b+1]})^T U^{[a+1,b+1]} + (\hat{\beta}^{[a+1,b+1]})^T S^{[a+1,b+1]} \hat{\beta}^{[a+1,b+1]}}{\sum_{j=a+1}^{b+1} n_j - \|\hat{\beta}^{[a+1,b+1]}\|_0} \right\}^{1/2}; \tag{2.14}$$

$$\hat{\tau}^{[a+1,b+1]}_{r} = \sqrt{m^{[a+1,b+1]}_{r}} / a^{[a+1,b+1]}_{1,r}, \tag{2.15}$$

where

$$V^{[a+1,b+1]} = V^{[a,b]} + V^{(b+1)} - V^{(a)} \quad \text{and} \quad m^{[a+1,b+1]}_{r} = m^{[a,b]}_{r} + m^{(b+1)}_{r} - m^{(a)}_{r}. \tag{2.16}$$

The process for constructing the confidence interval is summarized in Algorithm 2.

Algorithm 2 (1-α)-confidence interval estimation

```
1: Input: significance level $\alpha$, summary statistics $V^{[a,b]}$, $m_r^{[a,b]}$, $V^{(j)}$, and $m_r^{(j)}$,
   for $j = a, \dots, b$ and $r = 1, \dots, p$, and all the input of Algorithm 1.
2: Collect data batch: $D_{b+1} = \{y^{(b+1)}, X^{(b+1)}\}$;
3: Using Algorithm 1 to obtain the non-debiased estimator $\hat{\beta}^{[a+1,b+1]}$ and debiased estimator $\hat{\beta}_{de}^{[a+1,b+1]}$, and the auxiliary summary statistics.
4: Compute $V^{(b+1)} = (y^{(b+1)})^T y^{(b+1)}$ and $m_r^{(b+1)} = (\hat{z}_r^{(b+1)})^T \hat{z}_r^{(b+1)}$, and update
   $V^{[a+1,b+1]}$ and $m_r^{[a+1,b+1]}$ using (2.16);
5: Compute $\hat{\sigma}_\epsilon^{[a+1,b+1]}$ and $\hat{\tau}_r^{[a+1,b+1]}$ using (2.14) and (2.15);
6: Obtain $(1-\alpha)$-confidence interval estimators of $\beta_{0,r}$,
   $$\left(\hat{\beta}_{de,r}^{[a+1,b+1]} - \Phi^{-1}(1-\frac{\alpha}{2})\left(\hat{\sigma}_\epsilon^{[a+1,b+1]}\hat{\tau}_r^{[a+1,b+1]}\right), \hat{\beta}_{de,r}^{[a+1,b+1]} + \Phi^{-1}(1-\frac{\alpha}{2})\left(\hat{\sigma}_\epsilon^{[a+1,b+1]}\hat{\tau}_r^{[a+1,b+1]}\right)\right).$$
7: Store and Clear: Store $V^{[a+1,b+1]}$, $m_r^{[a+1,b+1]}$, $V^{(j)}$, and $m_r^{(j)}$, for $j =$
   $a+1, \dots, b+1$ and $r = 1, \dots, p$, and clear others.
8: Output: The $(1-\alpha)$-confidence interval estimator of $\beta_0$ in line 6.
```

3. Asymptotic Normality

In this section, we focus on the debiased estimator $\hat{\beta}_{de,r}^{[a,b]}$ defined in (2.10) and establish its asymptotic property. To that end, we first introduce the assumptions and notations.

Assumption 1. *We assume the following assumptions hold:*

(i) The row vectors of each data batch $X^{(j)}$ for $j = 1, 2, \dots$ are i.i.d. *sub-Gaussian random vectors with covariance matrix Σ;*

(ii) The smallest eigenvalue σ_{min} of Σ satisfiesand $0 < C < \sigma_{min}$, where M is some positive constant;

(iii) *The maximum diagonal element of* Σ *satisfies* $\max_j \Sigma_{j,j} = O(1)$*;*

(iv) *The error terms* $\epsilon_l^{(j)}$*'s for* $j = 1, 2, \dots$ *and* $l = 1, ..., n_j$ *are* i.i.d. *sub-Gaussian random variables with variance* σ^2 *and finite sub-Gaussian parameter.*

Define $\Theta = \Sigma^{-1}$ and its row sparsity level $s_r = \#\{k : \Theta_{r,k} \neq 0, r \leq k\}$. Define the sparsity level of the true parameter β_0 as $s_0 = \#\{j : \beta_{0,j} \neq 0\}$. For $r = 1, \dots, p$, define

$$\gamma_r := \underset{\gamma \in \mathbb{R}^{p-1}}{\arg\min}\ \mathbb{E}[\|x_r - X_{-r}\gamma\|_2^2]. \tag{3.17}$$

Recall the bias-correction step of Algorithm 1, (3.17) is actually the population version of (2.6). Regardless of the computational efficiency, the ideal estimator for γ_r in the *offline setting* is

$$\hat{\gamma}_r^{[a,b]} := \underset{\gamma \in \mathbb{R}^{p-1}}{\arg\min} \left\{ \frac{1}{\sum_{j=a}^{b} n_j} \sum_{j=a}^{b} \|x_r^{(j)} - X_{-r}^{(j)}\gamma\|_2^2 + \tilde{\lambda}_{[a,b]}\|\gamma\|_1 \right\}. \tag{3.18}$$

The next lemma shows that under certain conditions, the estimator in (2.6) is close to that in (3.18), which is critical in establishing the asymptotic normality of $\hat{\beta}_{de,r}^{[a,b]}$.

Lemma 1. *Suppose Assumption 1 holds,* $n_j \gtrsim s_r \log p$*,* $\lambda_r^{(j)} \asymp \sqrt{\frac{\log p}{n_j}}$ *in*

(2.6) *for* $j = 1, \ldots, b$, *and* $\tilde{\lambda}_{[a,b]} \asymp \sqrt{\frac{\log p}{\sum_{j=a}^{b} n_j}}$ *in* (3.18). *Then, for* $r = 1, \ldots, p$, *with probability at least 1-p*$^{-3}$, *the* $\hat{\gamma}_r^{(j)}$ *in* (2.6) *and the* $\hat{\gamma}_r^{[a,b]}$ *in* (3.18) *satisfies*

$$\|\hat{\gamma}_r^{(j)} - \hat{\gamma}_r^{[a,b]}\|_1 \lesssim s_r \sqrt{\frac{\log p}{n_j}}. \tag{3.19}$$

With Lemma 1 at hand, the next theorem provides the asymptotic properties of $\hat{\beta}_{de,r}^{[a,b]}$.

Theorem 1. *Suppose Assumption 1 holds,* $n_j \gtrsim s_r \text{log} p$, $\sum_{j=a}^{b} n_j \gtrsim s_0 \log p$, $\lambda_r^{(j)} \asymp \sqrt{\frac{\log p}{n_j}}$ *in* (2.6) *for* $j = 1, \ldots, b$, $r = 1, \ldots, p$, *and* $\lambda_{[a,b]} \asymp \sqrt{\frac{\log p}{\sum_{j=a}^{b} n_j}}$ *in* (2.1), $\tilde{\lambda}_{[a,b]} \asymp \sqrt{\frac{\log p}{\sum_{j=a}^{b} n_j}}$ *in* (3.18). *If*

$$\frac{s_0 s_r \log p \sum_{j=a}^{b} \sqrt{n_j}}{\sum_{j=a}^{b} n_j} = o_p(1) \ \ and \ \ \frac{(b-a) s_r^2 \log p}{\sum_{j=a}^{b} n_j} = o_p(1),$$

then for $r = 1, 2, \ldots, p$ *and large enough* $\sum_{j=a}^{b} n_j$, *we have*

$$(\hat{\tau}_r^{[a,b]})^{-1}(\hat{\beta}_{de,r}^{[a,b]} - \beta_{0,r}) := w_r^{[a,b]} + \Delta_r^{[a,b]},$$

where $(\hat{\tau}_r^{[a,b]})^{-1} \asymp \sqrt{\sum_{j=a}^{b} n_j}$, $w_r^{[a,b]} \sim \mathcal{N}(0, \sigma^2)$, *and* $\Delta_r^{[a,b]} = o_p(1)$. *This* *means*

$$(\hat{\tau}_r^{[a,b]})^{-1}(\hat{\beta}_{de,r}^{[a,b]} - \beta_{0,r}) \xrightarrow{d} \mathcal{N}(0, \sigma^2).$$

Theorem 1 shows that the debiased estimator is asymptotically unbiased and the convergence rate is of the standard order $(\sum_{j=a}^{b} n_j)^{-1/2}$. Theorem 1 also provides theoretical support for the confidence interval proposed in Algorithm 2.

4. Simulation

In this section, we evaluate the finite sample performance of the proposed algorithm dOnFL. In Section 4.1, we verify the effect of debiasing in dOnFL. In Section 4.2, we test the computational efficiency of dOnFL. In Section 4.3, we examine the robustness of dOnFL to the distribution of noise and covariates, and to the selection of tuning parameters.

The following three algorithms are generally compared:

- dOnFL: the proposed algorithm. The tuning parameter $\lambda_{[a,b]}$ in (2.1) is selected by minimizing the prediction error; see Remark 2. The tuning parameter $\lambda_r^{(b)}$ in (2.6) is selected using AIC. The step size η_b is set as the theoretical value; see Remark 1 for details.

- OnFL: the counterpart of dOnFL without debiasing (i.e., Stage II of Algorithm 1). The tuning parameters and step size are selected using the same procedures as in dOnFL.

- dLASSO: the offline counterpart of dOnFL, ; see Algorithm S.1. It consists with dOnFL in Stage I, but differs in Stage II by leveraging all available data batches $D_{[a,b]}$ up to time stamp b $(1 \leq b < \infty)$ to compute the bias term. Therefore, dLASSO can be regarded as the baseline method for statistical accuracy. The tuning parameters and step size are also selected using the same procedures as in dOnFL.

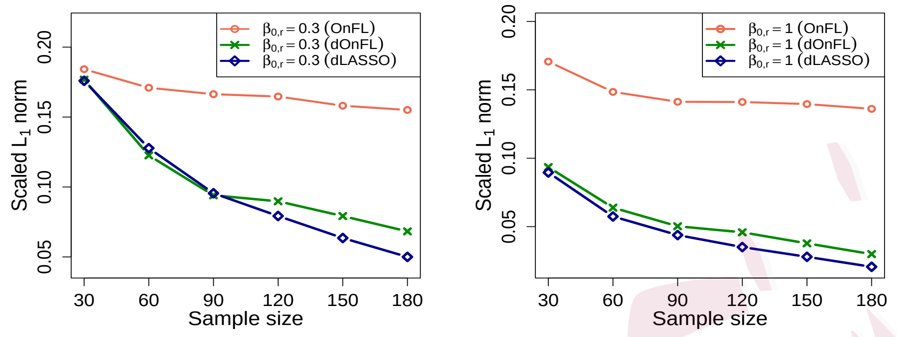
Experimental set-up: The general parameter settings are as follows, with some modifications in Section 4.3. We fix the number of avaliable batches at each time stamp as $T = b - a + 1 = 3$. We assume the identical sample size n_j's in each data batch size and let n_j vary in $\{10, 20, 30, 40, 50, 60\}$, that is, the number sample sizes in each time stamp vary in $\{30, 60, 90, 120, 150, 180\}$. The dimension of true parameters is $p = 200$. The true parameter sparsity level $s_0 = 10$, with $s_0/2$ strong signals $\beta_{0,s} = 1$ and $s_0/2$ weak signals $\beta_{0,w} = 0.3$. The variance of Gaussian noise is $\sigma = 0.3$. The covariates are generated from the zero-mean multivariate Gaussian distribution. We consider two set-up for the covariance matrices: set-up I: $\Sigma = \{0.4^{|i-j|}\}_{i,j=1,\ldots,p}$;

set-up II: $\Sigma = \mathbb{I}$. All the experiments througout the paper were performed on a laptop with Intel Core i7-1165G7 CPU 2.80GHz, 16GB memory, and 64-bit WS operating-system.

4.1 Bias evaluation

We validate the debiasing ability of the proposed method dOnFL and compare it with the non-debiased counterpart OnFL, as well as the offline counterpart dLASSO. To this end, we measure the *scaled* L_1 *norm* of three methods with respect to the strong signals and weak signals. Specifically, the scaled L_1 norm with respect to strong signal is defined as $\frac{\|\hat{\beta}_s - \beta_{0,s}\|_1}{\|\beta_{0,s}\|_1}$, where $\hat{\beta}_s$ denotes the estimated parameters of the strong signals. The average scaled L_1 norm of the estimators obtained at all the time stamps is reported. Similarly, we can define the scaled L_1 norm with respect to weak signal.

Figures 1 and 2 display the average results over 20 replications under the two set-ups for the covariance matrix, respectively. We have the following observations. First, dOnFL consistently outperform OnFL, showing the efficacy of debiasing. Second, dOnFL performs only slightly worse than or comparably to the baseline method dLASSO, despite using fewer samples for debiasing. Third, as expected, for both methods, the strong signals are



(a) Scaled L_1 norm of weak signals (b) Scaled L_1 norm of strong signals

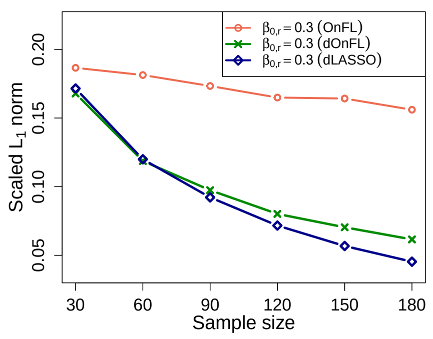
Figure 1: The performance of dOnFL,OnFL and dLASSO in terms of the scaled L_1 norm under the set-up I for the covariance matrix. The scaled L_1 norm is reported separately for strong and weak signals. The sample size represents the number of samples in the avaliable batches at each time stamp.

estimated more accurately than the weak signals.

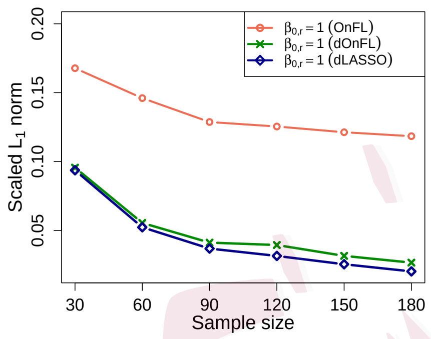
4.2 Computational efficiency evaluation

We evaluate the computational efficiency of dOnFL and compare it with the offline counterpart dLASSO. The goal is to test whether dOnFL enhances the computational efficiency with limited loss of the statistical accuracy.

Regarding the computational efficiency, we test the median running time (seconds) of two algorithms over 20 replications. Specifically, the running time is the total time over all time stamps as the number of sample size varies. Regrading the statistical accuracy, we test three measures. The first



(a) Scaled L_1 norm of weak signals



(b) Scaled L_1 norm of strong signals

Figure 2: The performance of dOnFL, OnFL and dLASSO in terms of the scaled L_1 norm under the set-up II for the covariance matrix. The scaled L_1 norm is reported separately for strong and weak signals. The sample size represents the number of samples in the avaliable batches at each time stamp.

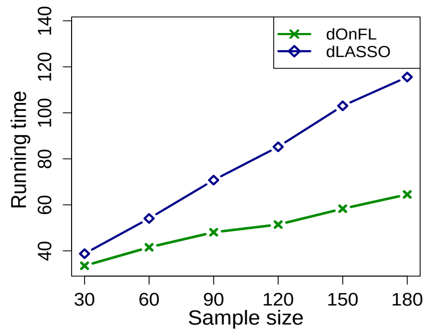
is the scaled L_1 norm defined in Section 4.1. The second is the length of the estimated confidence interval. The third is the coverage rate of the estimated confidence interval, namely, the probability of the estimated interval covering the true underlying parameter.

Figures 3 and 4 show the average (median for the running time) results over 20 replications under the two set-ups for the covariance matrix, respectively. We have the following observations. First, the proposed method dOnFL shows superior advantage over dLASSO in terms of the computational efficiency; see also the discussions for time complexity in Section 2. Second, the proposed method dOnFL is close to dLASSO in terms of statistical accuracy, though slightly inferior. In particular, compared with dLASSO, dOnFL achieves a shorter interval length at the expense of a lower coverage probability. And the scaled L_1 norm of two methods are comparable. Based on these observations, we conclude that dOnFL greatly enhances the computational efficiency without sacrificing the statistical accuracy too much.

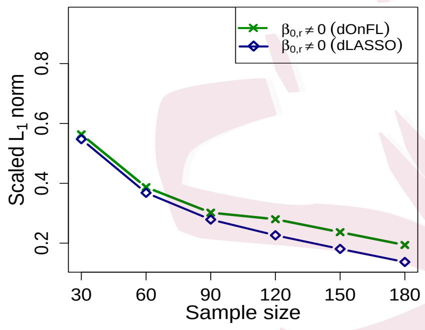
4.3 Sensitivity analysis and additional experiments

First, we test the robustness of dOnFL against the distribution of covariates and noise. Specifically, we consider several combinations of settings where the covariates follow Gaussian, Uniform, or t-distributions, and the noise follows Exponential, Uniform, or t-distributions. Apart from the distributional assumptions, the basic experimental setups are the same as described above. Table 1 presents the results when the covarites are Gaussian under set-up I and the noise terms are $i.i.d.$ $U(-0.5, 0.5)$. More results are regulated to Table S.2-S.6 in the Supplementary Materials. It turns out in all the considered set-ups, the dOnFL has advantage in computational efficiency over the dLASSO without sacrificing much statistical accuracy, which shows the robustness of the proposed algorithm.

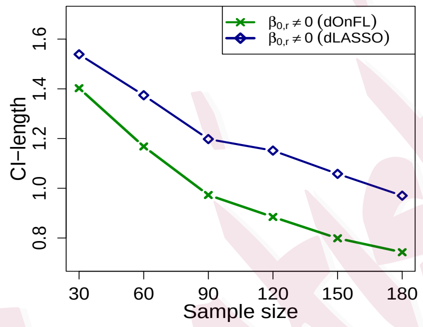
Second, we conduct experiments to compare the effect of three meth-



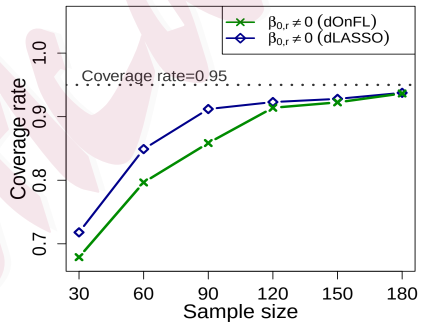
(a) Running time (seconds)



(b) Scaled L_1 norm

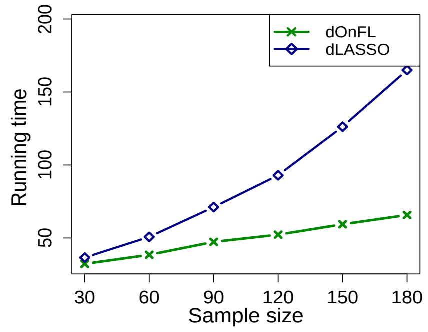


(c) Length of interval

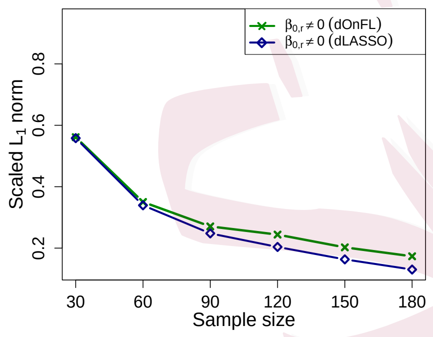


(d) Coverage rate

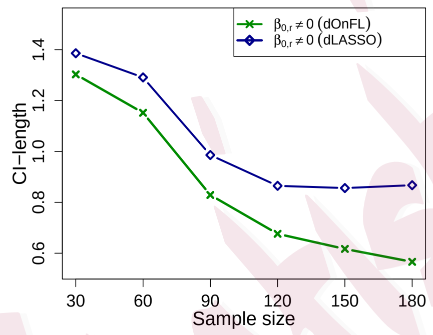
Figure 3: The performance of dOnFL and dLASSO in terms of the running time (seconds), scaled L_1 norm, length of interval and coverage rate under the set-up I for the covariance matrix.



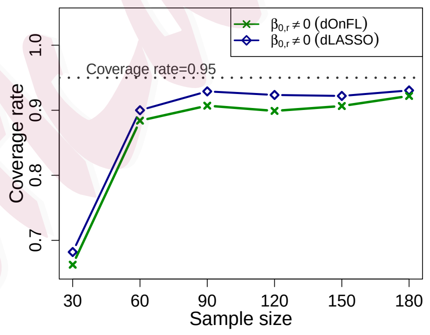
(a) Running time (seconds)



(b) Scaled L_1 norm



(c) Length of interval



(d) Coverage rate

Figure 4: The performance of dOnFL and dLASSO in terms of the running time (seconds), scaled L_1 norm, length of interval and coverage rate under the set-up II for the covariance matrix.

Table 1: The average performance of dOnFL, OnFL and dLASSO over 20 replications as the sample size (i.e., the number of samples in the available batches at each time stamp) increases. The error terms are *i.i.d.* $U(-0.5, 0.5)$ and the covariates for each sample are *i.i.d.* p-dimensional Gaussian $N(0, \Sigma)$ with $\Sigma_{ij} = 0.4^{|i-j|}$.

Performance Metric	Algorithm	Sample Size					
		30	60	90	120	150	180
Scaled L_1 norm of weak signals $\beta_{0,r} = 0.3$	OnFL	0.187	0.171	0.168	0.162	0.159	0.157
	dOnFL	0.184	0.122	0.101	0.081	0.081	0.071
	dLASSO	0.193	0.119	0.097	0.079	0.064	0.048
Scaled L_1 norm of strong signals $\beta_{0,r} = 1$	OnFL	0.175	0.151	0.142	0.138	0.135	0.141
	dOnFL	0.099	0.067	0.049	0.040	0.034	0.031
	dLASSO	0.097	0.060	0.042	0.032	0.025	0.022
CI-length	dOnFL	1.393	1.172	0.958	0.901	0.774	0.763
	dLASSO	1.522	1.386	1.192	1.162	1.022	0.993
Coverage rate	dOnFL	0.645	0.775	0.853	0.921	0.922	0.942
	dLASSO	0.670	0.833	0.911	0.930	0.930	0.943
Running time (seconds)	dOnFL	40.340	44.305	52.345	60.100	66.700	69.070
	dLASSO	46.745	58.800	77.245	97.660	117.415	127.160

ods for the tuning parameter selection in (2.6), namely, AIC, BIC, cross-validation (CV), on the statistical performance of three methods, namely, dOnFL, OnFL and dLASSO. The experimental set-up is the same with that introduced at the begining of this section with the identical covariance matrix. The results are regulated to Figure S.1 in the Supplementary Materials. The results demonstrate that all three methods exhibit robust performance regardless of the tuning parameter selection approach.

Third, apart from the tuning parameter in (2.6), we also test various step sizes in the ISTA algorithm for lasso to see the sensitivity of all methods. See the experimental details and results in Table S.7 in the Supplementary Materials. It turns out for all the tested step size, the proposed method dOnFL performs better than OnFL and comparable to dLASSO.

Finally, we conduct a higher dimensional experiment with ($p = 800$) and $T = b - a + 1 = 2$. We assume the identical sample size n_j's in each data batch size and let n_j vary in $\{80, 120, 160, 200, 240\}$, that is, the number sample sizes in each time stamp vary in $\{160, 240, 320, 400, 480\}$. The distribution set-up of the noise and covariates are the same with the set-up II introduced at the begining of this section. The results, summarized in Table S.8 in the Supplementary Materials, also show that the proposed dOnFL exhibits slightly lower statistical accuracy compared to dLASSO, but achieves significantly higher computational efficiency.

5. Real data analysis

In this section, we test the efficacy of dOnFL on three real datasets, Medical Cost Personal Dataset, California Housing Dataset and Blog Posts Dataset. To mimic the online-forgetting set-up, we divided each dataset into B total batches. New batch arrives sequentially and the oldest batch are removed. The details of each dataset and the corresponding parameter set-ups are as follows.

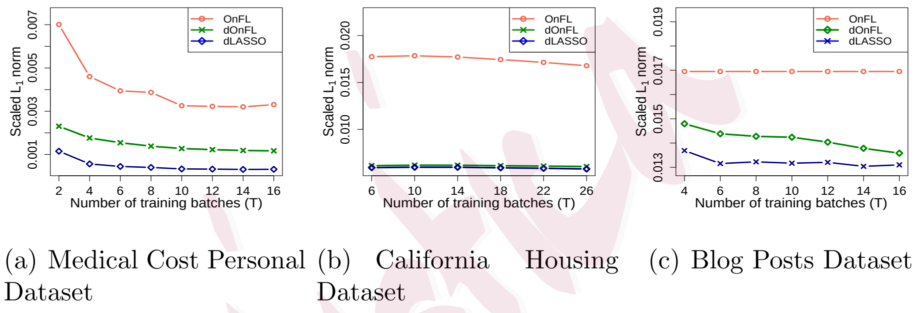
Medical Cost Personal Dataset. This dataset contains the age (discrete), sex (binary), body mass index (BMI) (continuous), number of children (discrete), smoking status (binary), region of residence (categorical with 4 classes) and medical cost (continuous) of 1,330 individuals. The goal is to predict the medical cost using other variables. The raw data can be found at https://www.kaggle.com/datasets/mirichoi0218/insurance. We equally divide the dataset into $B = 35$ batches.

California Housing Dataset. In this dataset, each samples represents a block group in California from the 1990 Census, comprising a total of 20,640 block groups. The goal is to predict the median house value within each block group using the covariates including median income, median age, total population, number of households, and the total number of rooms across all houses in the block group. The raw data is constructed by Pace and Barry (1997) and can be found at https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html. We equally divide the dataset into $B = 36$ batches.

Blog Posts Dataset. This dataset includes the number of comments received within 24 hours of a blog post being published, as well as 280 factors that may influence the number of comments. Each sample corresponds to a blog. The goal is to predict the number of comments using 280 variables include average, standard deviation, min, max and median of the length of time between the publication of the blog post and "current" time, the length of the blog post and so on. We consider the blog posts between February 1, 2012, and March 31, 2012, which results in 7,624 samples. The raw data can be downloaded from https://archive.ics.uci.edu/ml/datasets/BlogFeedback. We observe that various variables have many zeros which may affect the accuracy of prediction. Thus, before implementing the proposed algorithm, we conduct the lasso using the whole dataset. The number of selected variables results in 21. We treat the blog post corresponding to each day as one batch which results in $B = 59$ total batches.

For each of the three datasets, we evaluate the accuracy and efficiency of the proposed method dOnFL. In terms of the accuracy, we compare dOnFL with the non-debiased counterpart OnFL. We also compare the offline algorithm dLASSO, serving as the gold-standard for the accuracy of the online-forgetting algorithms. The details of algorithms can be found in Section 4. We report the average scaled L_1 norm over all the time stamps as the number of retained batches T vary. The norm is then averaged over 20 replications. The results for three datasets are shown in Figure 5. In terms of the efficiency, we compare dOnFL with the offline algorithm dLASSO. We report the median running time (seconds) over 20 replications as the time stamp increases. The results for three datasets are shown in Figure 6.

Similar to the simulation, from the results, we observe that the proposed dOnFL shows superior efficiency than the offline counterpart dLASSO while maintaining satisfactory statistical accuracy. In addition, the non-debiased OnFL performs worse than dOnFL, showing the efficacy of the debiasing technique.

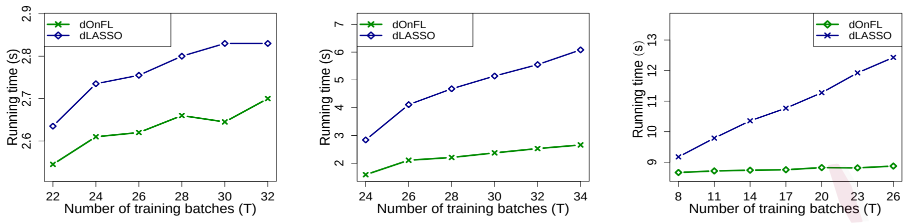


(a) Medical Cost Personal Dataset (b) California Housing Dataset (c) Blog Posts Dataset

Figure 5: The performance of dOnFL, OnFL, and dLASSO in terms of scaled L_1 norm on three real datasets.

6. Conclusion

In this paper, we considered the problem of data removal for statistical inference of lasso. To that end, we developed a new algorithm dOnFL within the online-forgetting framework, where the new data batches arrive sequen-



(a) Medical Cost Personal Dataset (b) California Housing Dataset (c) Blog Posts Dataset

Figure 6: The performance of dOnFL and dLASSO in terms of running time (seconds) on three real datasets.

tially while the earliest data batches are removed to maintain a constant memory constraint. In particular, we proposed a new debiasing technique to efficiently reduce the bias induced by lasso. The proposed algorithm dOnFL enjoys computational efficiency and statistical accuracy, and it is summary statistics-based. We established the asymptotic normality of dOnFL and provided the method for confidence interval estimation. Numerical experiments showed the advantage of dOnFL over several competitors.

There are many ways to extend the content of this paper. First, in our algorithm, the users' privacy was preserved to some extent by using the summary statistics instead of the raw data. It is meaningful to study the problem under a more rigorous privacy-preserving framework, e.g., the differential privacy (Dwork et al., 2006; Cai et al., 2023). Second, it would be interesting to consider other framework for data removal, e.g., the certified removal (Guo et al., 2020), for the high-dimensional regression problem. In addition, it is beneficial but challenging to develop the iterative algorithm for data removal just like the stochastic gradient descent algorithm for online learning (Han et al., 2024).

Acknowledgements

Our research is partially supported by National Natural Science Foundation of China (No.12301384, 12326615) and the Major Key Project of PCL under Grant PCL2024A06. The authors are grateful to the editor, associate editor, and two anonymous reviewers for their helpful and insightful comments.

References

Beck, A. and M. Teboulle (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences 2*(1), 183–202.

Bourtoule, L., V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot (2021). Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE.

Cai, Z., S. Li, X. Xia, and L. Zhang (2023). Private estimation and inference in high-dimensional regression with fdr control. *arXiv preprint arXiv:2310.16260*.

Chen, X., J. D. Lee, X. T. Tong, and Y. Zhang (2020). Statistical inference for model parameters in stochastic gradient descent. *Annals of Statistics 48*, 251–273.

Deshpande, Y., A. Javanmard, and M. Mehrabi (2023). Online debiasing for adaptively collected high-dimensional data with applications to time series analysis. *Journal of the American Statistical Association 118*(542), 1126–1139.

Duchi, J., E. Hazan, and Y. Singer (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*(7).

Dwork, C., F. McSherry, K. Nissim, and A. Smith (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pp. 265–284. Springer.

Friedman, J., T. Hastie, H. Höfling, and R. Tibshirani (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics 1*(2), 302–332.

Ginart, A., M. Guan, G. Valiant, and J. Y. Zou (2019). Making ai forget you: Data deletion in machine learning. *Advances in Neural Information Processing Systems 32*.

Guo, C., T. Goldstein, A. Hannun, and L. Van Der Maaten (2020, 13–18 Jul). Certified data removal from machine learning models. In H. D. III and A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, Volume 119 of *Proceedings of Machine Learning Research*, pp. 3832–3842. PMLR.

Han, R., L. Luo, Y. Lin, and J. Huang (2024). Online inference with debiased stochastic gradient descent. *Biometrika 111*(1), 93–108.

Izzo, Z., M. A. Smart, K. Chaudhuri, and J. Zou (2021). Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pp. 2008–2016. PMLR.

Javanmard, A. and A. Montanari (2014). Confidence intervals and hypothesis testing for high-dimensional regression. *Journal of Machine Learning Research 15*(1), 2869–2909.

Langford, J., L. Li, and T. Zhang (2009). Sparse online learning via truncated gradient. *Journal of Machine Learning Research 10*(3).

Li, Y., C.-H. Wang, and G. Cheng (2021). Online forgetting process for linear regression models. In *International Conference on Artificial Intelligence and Statistics*, pp. 217–225. PMLR.

Liu, X. and S. A. Tsaftaris (2020). Have you forgotten? a method to assess if machine learning models have forgotten data. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8,* *2020, Proceedings, Part I 23*, pp. 95–105. Springer.

Pace, R. K. and R. Barry (1997). Sparse spatial autoregressions. *Statistics & Probability Letters 33*(3), 291–297.

Parikh, N., S. Boyd, et al. (2014). Proximal algorithms. *Foundations and trends® in Optimization 1*(3), 127–239.

Shi, C., R. Song, W. Lu, and R. Li (2021). Statistical inference for high-dimensional models via recursive online-score estimation. *Journal of the American Statistical Association 116*(535), 1307–1318.

Sun, L., M. Wang, S. Zhu, and A. Barbu (2024). A novel framework for online supervised learning with feature selection. *Journal of Nonparametric Statistics*, 1–27.

Tarres, P. and Y. Yao (2014). Online learning as stochastic approximation of regularization paths: Optimality and almost-sure convergence. *IEEE Transactions on Information Theory 60*(9), 5716–5735.

van de Geer, S., P. Bühlmann, Y. A. Ritov, and R. Dezeure (2014). On asymptotically optimal confidence regions and tests for high-dimensional models. *Annals of Statistics 42*, 1166–1202.

Zhang, C.-H. and S. S. Zhang (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society Series B: Statistical Methodology 76*(1), 217–242.

Xiao Guo, School of Mathematics, Northwest University, Xi'an, China

Xu Zhang, School of Mathematics, Northwest University, Xi'an, China

Hai Zhang, School of Mathematics, Northwest University, Xi'an, China

E-mail: zhanghai@nwu.edu.cn