Chunk Dynamic Updating for Group Lasso with ODEs

Diyang Li[1], **Bin Gu**[2,1]

[1] School of Computer & Software, Nanjing University of Information Science & Technology, P.R.China
[2] MBZUAI, United Arab Emirates
DiyoungLee@gmail.com, bin.gu@mbzuai.ac.ae

Abstract

Group Lasso is an important sparse regression method in machine learning which encourages selecting key explanatory factors in a grouped manner because of the use of $\ell_{2,1}$-norm. In real-world learning tasks, some chunks of data would be added into or removed from the training set in sequence due to the existence of new or obsolete historical data, which is normally called dynamic or lifelong learning scenario. However, most of existing algorithms of group Lasso are limited to offline updating, and only one is online algorithm which can only handle newly added samples inexactly. Due to the complexity of $\ell_{2,1}$-norm, how to achieve accurate chunk incremental and decremental learning efficiently for group Lasso is still an open question. To address this challenging problem, in this paper, we propose a novel accurate dynamic updating algorithm for group Lasso by utilizing the technique of Ordinary Differential Equations (ODEs), which can incorporate or eliminate a chunk of samples from original training set without retraining the model from scratch. Specifically, we introduce a new formulation to reparameterize the adjustment procedures of chunk incremental and decremental learning simultaneously. Based on the new formulation, we propose a path following algorithm for group Lasso regarding to the adjustment parameter. Importantly, we prove that our path following algorithm can exactly track the piecewise smooth solutions thanks to the technique of ODEs, so that the accurate chunk incremental and decremental learning can be achieved. Extensive experimental results not only confirm the effectiveness of proposed algorithm for the chunk incremental and decremental learning, but also validate its efficiency compared to the existing offline and online algorithms.

1 Introduction

Group Lasso (Yuan and Lin 2006) considers the problem of selecting grouped variables for explanatory prediction in regression, which has been successful in many practical applications of machine learning (Ma, Song, and Huang 2007; Chatterjee et al. 2012; Rao et al. 2015; Zhang et al. 2019; Huo et al. 2020) and statistical analysis (Yuan and Lin 2006; Bach 2008; Lim and Hastie 2015). The utilizing of group Lasso penalty (also called $\ell_{2,1}$-norm penalty) leads coefficient that only contains a few of wanted groups rather than sparsity in individual elements, which can be viewed as an important extension of Lasso (Tibshirani 1996) model. Considering attractive features and their good generalization performance, group Lasso have received considerable interest in machine learning community and has been intensively studied. Some relevant models include group Lasso for logistic regression (Meier, Van De Geer, and Bühlmann 2008), overlapped group Lasso (Jacob, Obozinski, and Vert 2009) and sparse group Lasso (Simon et al. 2013). More than that, (Casella et al. 2010) proposed a fairly general fully Bayesian formulation which could accommodate various Lasso variations (e.g. Bayesian group Lasso), and (Simon and Tibshirani 2012) put up the standardized group Lasso.

In lots of real-world application scenarios, e.g., cloud computing (Armbrust et al. 2010) and federated learning system (McMahan et al. 2017), the new chunk of data is continuously being generated at unpredictable rates and comes in different sizes. One way to handle the coming data stream is to perform online training as new samples come. Therefore, the training of a system must be conducted sequentially in online fashion (also called dynamic updating). More than that, to train in a dynamic scene, we are demanded to drop some wrong or obsolete historical data from the existing set. Thus incremental and decremental learning (Giraud-Carrier 2000; Gepperth and Hammer 2016) are very important learning paradigm, because the model can refine its knowledge without re-training from scratch.

Nowadays most of existing algorithms for group Lasso are trained offline in batch-mode (Ida, Fujiwara, and Kashima 2019; Zhang et al. 2020), which is not applicable for real-time scenario, thus developing an efficient algorithm that copes with input data supplied in sequence is a desirable but challenging task for researchers. The online optimization framework (Zinkevich 2003; Bottou and LeCun 2004) is an important learning system to handle incremental learning, but to the best of our knowledge, existing online algorithms for group Lasso, DA-GL (Yang et al. 2010) and ADA-GL (Li et al. 2014) give an inexact solution and has been confined to incremental updating only. Some related works are summarized in Table 1. Due to the complexity of $\ell_{2,1}$-norm, it is still unknown how to achieve accurate chunk incremental and decremental learning efficiently for group Lasso.

To fill this gap, in this paper we contribute with a novel updating algorithm about group Lasso for a chunk of new samples, Chunk Incremental (Decremental) Group Lasso

Table 1: Representative dynamic updating algorithms for (group) Lasso.

Problem	Reference	Exact	Incremental	Decremental	Chunk
Lasso	(Duchi and Singer 2009; Xiao 2010)	No	Yes	No	No
Lasso	(Garrigues and Ghaoui 2008)	Yes	Yes	Yes	No
group Lasso	(Yang et al. 2010)	No	Yes	No	No
group Lasso	(Li et al. 2014)	No	Yes	No	Incremental only
group Lasso	Our	**Yes**	**Yes**	**Yes**	**Yes**

Algorithm, namely "CIGL" ("CDGL"), where the updated model is exactly the same as a model trained from scratch using the entire dataset. Specifically, we introduce a new function to reparameterize the adjustment procedure, based on its new variable θ we build the difference equation and rewrite it with the Taylor expansion of infinite order to build an ODE, so as to compute the solution path as the adjustment is taking place. The idea of solution paths is to compute a compact representation of all optimal solutions, the optimal solution set forms a number of piecewise smooth curves in the solution space (Rosset and Zhu 2007). The Figure 1 shows an example of the piecewise smoothness for the solutions of group Lasso with respect to successively varied θ.

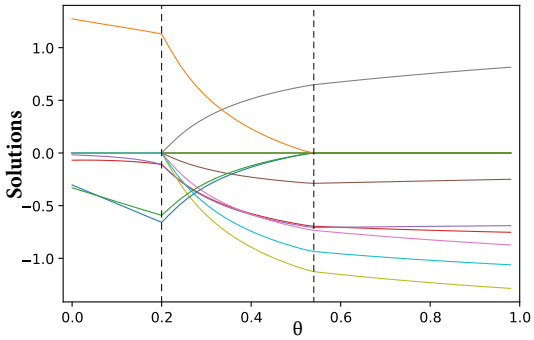


Figure 1: Piecewise smooth path of θ when adjusting one new sample, where one piecewise smooth curve corresponds to one variable in group Lasso.

Hence, our provably accurate algorithm can be applied to both incremental and decremental learning without retraining a model from scratch. The new algorithm enjoys high efficiency as it only requires to solve the first-order ODEs in most cases. The main contributions of this paper are summarized as follows.

- We design a novel framework for dynamic updating without retraining the model on whole dataset, where the dynamic changes can include both addition and deletion of chunk of samples.
- We are the first to propose a chunk incremental and decremental algorithm for group Lasso. Our experiments reveal that proposed algorithm has competitive accuracy, while being significantly faster than conventional batch training.
- In this paper we prove its accuracy and assess the computational cost and application limitations. Moreover, we provide theoretical guidance to its flexible extension.

2 Formulation of Dynamic Group Lasso

In this part, we introduce our learning framework of dynamic updating for group Lasso when incorporating or removing data.

2.1 A Revisit of Group Lasso

Given the dataset $X \in \mathbb{R}^{n\times m}$ with n observations and label vector $\boldsymbol{y} \in \mathbb{R}^n$, we assume that $\boldsymbol{y}$ is centered, i.e., $\sum_{i=1}^n y_i = 0$, and each feature of the training set X is standardized. Let $\boldsymbol{w}_g$ be the partitioning of $\boldsymbol{w}$ according to the g-th group. We consider a group Lasso regression model without the intercept, which is expressed as

$$\min_{\boldsymbol{w}} \frac{1}{2}\|X\boldsymbol{w}-\boldsymbol{y}\|^2 + \alpha \sum_{g=1}^{G} \sqrt{d_g}\,\|\boldsymbol{w}_g\|_2 . \tag{1}$$

There has a total of G groups. d_g is the number of elements in $\boldsymbol{w}_g$. The $\alpha > 0$ is the regularization parameter to balance the prediction loss and $\ell_{2,1}$ regularization term. We define the active set $\mathcal{A}$ that contains indices of the groups of $\boldsymbol{w}$ that $\boldsymbol{w}_g \neq 0$, so as to store active components or groups, i.e., we have $\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_{\mathcal{A}} \\ \boldsymbol{w}_{\overline{\mathcal{A}}} \end{bmatrix}$, where $\boldsymbol{w}_{\overline{\mathcal{A}}}$ remains 0.

Remark 1 *Consider $d_g = 1$ for all g, group Lasso (1) reduces to the ordinary Lasso model.*

2.2 Formulation of Dynamic Updating

The proposed Updating Factor in Objective (UFO) is formally defined as follows.

Definition 1 (UFO) *A rational ϕ_θ can be used to reparameterize the addition or subtraction procedure of a chunk data $X_{new} \in \mathbb{R}^{\mathcal{N}\times m}$, which satisfies the following conditions:*
1. ϕ_θ is monotonically increasing function w.r.t. θ in closed interval $[0, 1]$.
2. ϕ_θ has the properties of continuity and smoothness, meanwhile $\phi_\theta|_{\theta=0} = 0$, $\phi_\theta|_{\theta=1} = 1$ should be kept.

Through fusing ϕ_θ into the learning framework, we can make dynamic updating about X_{new}. Firstly, consider adding one chunk data, vary θ from 0 to 1 in

$$\min_{\boldsymbol{w}} \frac{1}{2}\left\| \begin{pmatrix} X \\ \phi_\theta X_{new} \end{pmatrix} \boldsymbol{w} - \begin{pmatrix} \boldsymbol{y} \\ \phi_\theta \boldsymbol{y}_{new} \end{pmatrix} \right\|_F^2 + \alpha \mathcal{R}_\Omega(\boldsymbol{w}), \tag{2}$$

where $\mathcal{R}_\Omega(\cdot)$ is the regularization term in some specific machine learning tasks, and ϕ_θ is the UFO defined in Definition 1, which formulates the processing of taking the newly

added samples into the training set. The $X_{new} \in \mathbb{R}^{\mathcal{N}\times m}$ and $\boldsymbol{y}_{new} \in \mathbb{R}^{\mathcal{N}}$ are incoming chunk data with $\mathcal{N}$ samples and their label, respectively. Here we are supposed to follow the solution path of $\boldsymbol{w}$ as θ increasing until we gain $\boldsymbol{w}$ at $\theta = 1$. Conversely, as we vary θ from 1 to 0, the decremental learning can then be performed in a similar way.

In Lasso problem, an homotopy algorithm (Garrigues and Ghaoui 2008) for updating one sample each time has been developed under the assumption of $\phi_\theta \overset{\text{def}}{=} \theta$. Taking the stationarity condition of Lasso, $\boldsymbol{w}_{\mathcal{A}}$ can be obtained in the closed form. However, it's non-trivial in group Lasso for analytical solutions.

3 Path Following via ODEs

In this section, we develop an efficient algorithm for solving the solution path w.r.t. θ in (2).

3.1 Optimality Conditions

In particular, we select transformation as $\phi_\theta \overset{\text{def}}{=} \sqrt{\theta}$ in (2) to derive chunk incremental and decremental group Lasso. Letting $\mathcal{C}_g \overset{\text{def}}{=} X_g^T(\boldsymbol{y} - X\boldsymbol{w}) + \theta X_{\text{new},g}^T(\boldsymbol{y}_{\text{new}} - X_{\text{new}}\boldsymbol{w})$, where X_g holds the columns of g-th group, the KKT conditions for group Lasso problem are formulated as follows.

$$\begin{aligned} \mathcal{C}_g &= \frac{\alpha\sqrt{d_g}\cdot \boldsymbol{w}_g}{\|\boldsymbol{w}_g\|} && \text{if } g \in \mathcal{A}, \\ \mathcal{C}_g &= \alpha\sqrt{d_g}\tau_g, \tau_g \in \{\boldsymbol{\tau}_g \big| \, \|\boldsymbol{\tau}_g\|_2 \leqslant 1\} && \text{if } g \in \overline{\mathcal{A}}. \end{aligned} \tag{3}$$

We denote $\mathcal{R}(\boldsymbol{w}) = \sum_g \sqrt{d_g}\,\|\boldsymbol{w}_g\|_2$, $\widetilde{X} = \begin{bmatrix} X \\ \sqrt{\theta}X_{new} \end{bmatrix}$, $\widetilde{y} = \begin{pmatrix} \boldsymbol{y} \\ \sqrt{\theta}\boldsymbol{y}_{new} \end{pmatrix}$ and $\mathcal{G}_\theta(\boldsymbol{w}) = \frac{1}{2}\left\|\widetilde{X}\boldsymbol{w} - \widetilde{y}\right\|_F^2 + \alpha\mathcal{R}(\boldsymbol{w})$.

3.2 Piecewise Smooth Path

Let $\boldsymbol{w}^*(\theta)$ be the minimizer of $\mathcal{G}_\theta(\boldsymbol{w})$. The following proposition guarantees the continuity of solution path regarding to θ.

Proposition 1 *The solution path $\boldsymbol{w}^*(\theta)$ is continuous of θ, where $\theta \in [0, 1]$.*

The detailed proof is provided in Appendix A. Further, inspired by the theoretical analysis in (Zhou and Wu 2014) and (Yau and Hui 2017), we derive a system of ODEs to compute the solution path w.r.t. θ. Note that this paper is intrinsically different from (Yau and Hui 2017), because we do not focus on optimal regularization parameter, but on an artificially parameter θ to conduct dynamic updating.

Theorem 1 *Given a fixed set $\mathcal{A}$, the exact solution path $\boldsymbol{w}^*(\theta)$ satisfies the following first-order ODE system:*

$$\frac{d\boldsymbol{w}_{\mathcal{A}}^*}{d\theta} = -\left[\widetilde{X}_{\mathcal{A}}^T\widetilde{X}_{\mathcal{A}} + \alpha\nabla^2\mathcal{R}\right]^{-1} X_{new,\mathcal{A}}^T\widetilde{\delta}, \tag{4}$$

where $\widetilde{X}_{\mathcal{A}}^T\widetilde{X}_{\mathcal{A}} = \theta X_{new,\mathcal{A}}^T X_{new,\mathcal{A}} + X_{\mathcal{A}}^T X_{\mathcal{A}}$ and $\widetilde{\delta} = X_{new,\mathcal{A}}\boldsymbol{w}_{\mathcal{A}}^*(\theta) - \boldsymbol{y}_{new}$.

We defer the proof of Theorem 1 in Section 4. By solving the initial value problem (4) numerically with ODE solvers, the solution path regarding to θ can be computed swiftly before $\mathcal{A}$ changes. We denote such a point where the active set changes a "transition point". The path would be smooth before θ approaches a transition point, and be non-smooth graphically at transition points.

Transition Point The emergence of a transition point is owing to indices change in $\mathcal{A}$, which is characterized by the variation w.r.t. the norm value of $\mathcal{C}_g$. Meanwhile the ODE wouldn't work correctly due to they are non-differentiable points.

As it's evident in optimality conditions, a non-zero $\boldsymbol{w}_g$ will decay to $\mathbf{0}$ as $\|\mathcal{C}_g\| \leqslant \alpha\sqrt{d_g}$ is found, meanwhile, index g is removed from set $\mathcal{A}$. In our implementation, based on the continuity of solutions given by an ODE solver, we can set $\boldsymbol{w}_g = \mathbf{0}$ $(g \in \mathcal{A})$ as soon as it reaches the opposite sign (i.e. path pass through 0).

Alternatively, when the norm value of $\mathcal{C}_g (g \in \overline{\mathcal{A}})$ comes to $\alpha\sqrt{d_g}$, the g-th group becomes active and will be added into $\mathcal{A}$. (3) indicates that $\mathcal{C}_g (g \in \mathcal{A})$ shares the collinearity with $\boldsymbol{w}^g$. Therefore an extreme short syntropic coefficient vector $\epsilon\mathcal{C}_g$ may be used as an approximation near transition point, which will give a start value for the following ODE solver, where ϵ is a user-defined tuning parameter and only lead into a controllable approximation. Note that $\mathcal{C}_g$ has the property of continuity, but is non-monotonic about θ. With that, we can keep observation on the sign and value of $\|\mathcal{C}_g\| - \alpha\sqrt{d_g}$ for each g in $\overline{\mathcal{A}}$ while solving the solution path to estimate whether there exists a transition point potentially.

3.3 Detailed Algorithm

In the updating process, we track the solutions with regard to θ. After detecting a new turning point, we need to reset $\mathcal{A}$ and recompute solutions. The above procedure is repeated until we traverse the entire interval $[0, 1]$. We show the detailed updating algorithm in Algorithm 1.

Remark 2 *Note if the whole path is smooth on $[0, 1]$, algorithm needn't cost extra time on detecting specific locations and recalculating the solution path, which apparently reduces the computation complexity when there are no transition points.*

Acceleration Technique Based on our empirical experience, in most updating processes, there have few changes on active set. Thus we can firstly solve $\boldsymbol{w}$ using (4) with the presumption of there are no turning points. Meanwhile we have to test whether our hypothesis is correct at the end of the interval.

Proposition 2 *We assume that the data matrix $X_{\mathcal{A}}$ is linearly independent w.r.t. its columns, then $\theta X_{new,\mathcal{A}}^T X_{new,\mathcal{A}} + X_{\mathcal{A}}^T X_{\mathcal{A}} + \alpha\nabla^2\mathcal{R}$ is a real symmetric positive definite matrix, where $\theta \in [0, 1]$.*

We provide the detailed proof in our Appendix B. During the derivation of (4), we essentially solved a linear system.

The above proposition guarantees that Cholesky decomposition can be utilized in the computations to raise the efficiency (Golub and Van Loan 2013). The linearly independence in Proposition 2 is easily guaranteed in a majority of real-world datasets and tasks, especially the sample size is relatively large. It's an optional optimization that can be used when the preconditions of no linear dependence can be ensured a priori, and we didn't use it (along with acceleration) in each test for fairness.

Algorithm 1: Chunk Incremental (Decremental) Group Lasso

Input: Initial solution $\boldsymbol{w}_0$, X, $\boldsymbol{y}$, X_{new}, $\boldsymbol{y}_{new}$
Output: Optimal $\boldsymbol{w}$

Set $\mathcal{A}$ according to $\boldsymbol{w}_0$, $\theta = 0$ (or $\theta = 1$ for decremental).
Partition $\boldsymbol{w}_{\mathcal{A}}$, $X_{\mathcal{A}}$, $X_{new,\mathcal{A}}$ by $\mathcal{A}$.
Solve (4) at $\theta = 1$ (or $\theta = 0$ for decremental).
if (3) was not met **then**
  $\theta = 0$ (or $\theta = 1$ for decremental).
  while $\theta \leq 1$ (or ≥ 0 for decremental) **do**
    Solve (4) and detect transition point simultaneously.
    if g-th group turns to inactive **then**
      $\boldsymbol{w}_g = \mathbf{0}$.
      Remove g from $\mathcal{A}$.
    else if g-th group becomes active **then**
      $\boldsymbol{w}_g = \epsilon \mathcal{C}_g$.
      Put g into $\mathcal{A}$.
    end if
    Update $\boldsymbol{w}_{\mathcal{A}}$, $X_{\mathcal{A}}$ and $X_{new,\mathcal{A}}$ according to the updated $\mathcal{A}$.
  end while
end if

4 Theoretical Analysis

In this section, we first provide the theoretical proof to Theorem 1, then give the complexity analysis of Algorithm 1.

4.1 Derivation of ODEs in Theorem 1

Recall $\mathcal{G}_\theta(\boldsymbol{w}) = \frac{1}{2}\left\|\widetilde{X}\boldsymbol{w} - \widetilde{y}\right\|_F^2 + \alpha\mathcal{R}(\boldsymbol{w})$. The KKT point of the original problem can be obtained by $\boldsymbol{w}^*(\theta) = \arg\min_{\boldsymbol{w}}\{\mathcal{G}_\theta(\boldsymbol{w})\}$. With a change of $\Delta\theta$, we can calculate the difference of the optimal $\boldsymbol{w}^*$, which implies $\boldsymbol{w}^*(\theta + \Delta\theta) - \boldsymbol{w}^*(\theta)$, as follows:

$$\Delta\boldsymbol{w}^* = \arg\min_{\Delta\boldsymbol{w}} \left\{\mathcal{G}_{\theta+\Delta\theta}\left(\boldsymbol{w}^*(\theta) + \Delta w\right) - \mathcal{G}_\theta\left(\boldsymbol{w}^*(\theta)\right)\right\} \overset{\text{def}}{=} \arg\min_{\Delta\boldsymbol{w}}\{\mathcal{D}\}.$$

By discarding the zero component (i.e. only pay attention to active set $\mathcal{A}$), and for convenience we define $\widetilde{E} = X_{\mathcal{A}}\boldsymbol{w}^*_{\mathcal{A}}(\theta) - \boldsymbol{y}$. We reset $\mathcal{R}(\boldsymbol{w}) = \sum_{g\in\mathcal{A}}\sqrt{d_g}\left\|\boldsymbol{w}^g\right\|_2$. The $\mathcal{G}_\theta(\boldsymbol{w})$ can be expanded as

$$\mathcal{G}_\theta(\boldsymbol{w}) = \frac{1}{2}\widetilde{E}^T\widetilde{E} + \frac{1}{2}\theta\cdot\widetilde{\delta}^T\widetilde{\delta} + \alpha\mathcal{R}(\boldsymbol{w}), \tag{5}$$

from which will be substituted into the $\mathcal{D}$. After simplification and reorganization, we obtain

$$\begin{aligned}\mathcal{D} = &\frac{1}{2}\Delta\boldsymbol{w}_{\mathcal{A}}^T\left[(\theta+\Delta\theta)X_{new,\mathcal{A}}^T X_{new,\mathcal{A}} + X_{\mathcal{A}}^T X_{\mathcal{A}}\right]\Delta\boldsymbol{w}_{\mathcal{A}} \\ &+ \left[(\theta+\Delta\theta)\widetilde{\delta}^T X_{new,\mathcal{A}} + \widetilde{E}^T X_{\mathcal{A}}\right]\Delta\boldsymbol{w}_{\mathcal{A}} \\ &+ \alpha\left[\mathcal{R}\left(\boldsymbol{w}^*_{\mathcal{A}}(\theta) + \Delta\boldsymbol{w}_{\mathcal{A}}\right) - \mathcal{R}\left(\boldsymbol{w}^*_{\mathcal{A}}(\theta)\right)\right] + \frac{1}{2}\widetilde{\delta}^T\widetilde{\delta}\cdot\Delta\theta.\end{aligned}$$

With utilization of the Taylor series expansion on $\mathcal{R}\left(\boldsymbol{w}^*_{\mathcal{A}}(\theta) + \Delta\boldsymbol{w}_{\mathcal{A}}\right)$ at the value of $\boldsymbol{w}^*_{\mathcal{A}}(\theta)$, the $\mathcal{D}$ turns into:

$$\begin{aligned}\mathcal{D} = &\frac{1}{2}\Delta\boldsymbol{w}_{\mathcal{A}}^T\left[(\theta+\Delta\theta)X_{new,\mathcal{A}}^T X_{new,\mathcal{A}} + X_{\mathcal{A}}^T X_{\mathcal{A}}\right]\Delta\boldsymbol{w}_{\mathcal{A}} \\ &+ \left[(\theta+\Delta\theta)\widetilde{\delta}^T X_{new,\mathcal{A}} + \widetilde{E}^T X_{\mathcal{A}}\right]\Delta\boldsymbol{w}_{\mathcal{A}} \\ &+ \alpha\cdot\left[(\nabla\mathcal{R})^T\Delta\boldsymbol{w}_{\mathcal{A}} + \frac{1}{2}\Delta\boldsymbol{w}_{\mathcal{A}}^T\left(\nabla^2\mathcal{R}\right)\Delta\boldsymbol{w}_{\mathcal{A}} + \right. \\ &\left.\sum_{p=3}^{\infty}\frac{1}{p!}\left(\nabla^p\mathcal{R}\right)\left(\Delta\boldsymbol{w}_{\mathcal{A}}\right)^p\right] + \frac{1}{2}\widetilde{\delta}^T\widetilde{\delta}\cdot\Delta\theta\end{aligned}$$

$$\begin{aligned}= &\frac{1}{2}\Delta\boldsymbol{w}_{\mathcal{A}}^T\left[(\theta+\Delta\theta)X_{new,\mathcal{A}}^T X_{new,\mathcal{A}} + X_{\mathcal{A}}^T X_{\mathcal{A}} + \alpha\nabla^2\mathcal{R}\right]\cdot \\ &\Delta\boldsymbol{w}_{\mathcal{A}} + \left[(\theta+\Delta\theta)\widetilde{\delta}^T X_{new,\mathcal{A}} + \widetilde{E}^T X_{\mathcal{A}} + \alpha(\nabla\mathcal{R})^T\right]\Delta\boldsymbol{w}_{\mathcal{A}} \\ &+ \frac{1}{2}\widetilde{\delta}^T\widetilde{\delta}\cdot\Delta\theta + \alpha\sum_{p=3}^{\infty}\frac{1}{p!}\left(\nabla^p\mathcal{R}\right)\left(\Delta\boldsymbol{w}_{\mathcal{A}}\right)^p,\end{aligned}$$

where $\nabla^p\mathcal{R}(p \geq 1)$ is coefficient matrix in high-dimensional space. Specifically, $\nabla^2\mathcal{R} = diag(B_1, \cdots, B_k)$ is a block diagonal matrix that the off-diagonal elements are 0, where $B_g = \frac{\sqrt{d_g}\left(\|\boldsymbol{w}^g\|^2\mathcal{I} - \boldsymbol{w}^g\boldsymbol{w}^{g\top}\right)}{\|\boldsymbol{w}^g\|^3}$, $g \in \mathcal{A}$, where $\mathcal{I}$ denotes identity matrix. As mentioned before, $\Delta\boldsymbol{w}^* = \arg\min_{\Delta\boldsymbol{w}}\mathcal{D}$, according to the stationarity condition (Boyd, Boyd, and Vandenberghe 2004), we get:

$$\begin{aligned}&\frac{\partial\mathcal{D}}{\partial(\Delta\boldsymbol{w}_{\mathcal{A}})} = \alpha\sum_{p=3}^{\infty}\frac{1}{(p-1)!}\left(\nabla^p\mathcal{R}\right)'\left(\Delta\boldsymbol{w}^*_{\mathcal{A}}\right)^{p-1} + \\ &\left[(\theta+\Delta\theta)X_{new,\mathcal{A}}^T X_{new,\mathcal{A}} + X_{\mathcal{A}}^T X_{\mathcal{A}} + \alpha\nabla^2\mathcal{R}\right]\Delta\boldsymbol{w}^*_{\mathcal{A}} + \\ &(\theta+\Delta\theta)X_{new,\mathcal{A}}^T\widetilde{\delta} + X_{\mathcal{A}}^T\widetilde{E} + \alpha\nabla\mathcal{R} = \mathbf{0},\end{aligned} \tag{6}$$

where $(\cdot)'$ is derivative note. Then, we rewrite (3) as:

$$X_{\mathcal{A}}^T\widetilde{E} + \theta X_{new,\mathcal{A}}^T\widetilde{\delta} + \alpha\nabla\mathcal{R} = \mathbf{0}. \tag{7}$$

Consequently, those terms vanish in (6). We now construct the difference equation about θ:

$$\underbrace{\alpha\sum_{p=3}^{\infty}\frac{\left(\nabla^p\mathcal{R}\right)'}{(p-1)!}\cdot\frac{\left(\Delta\boldsymbol{w}^*_{\mathcal{A}}\right)^{p-1}}{\Delta\theta}}_{\Upsilon_p} + \left[(\theta+\Delta\theta)X_{new,\mathcal{A}}^T X_{new,\mathcal{A}} + \right.$$
$$\left. X_{\mathcal{A}}^T X_{\mathcal{A}} + \alpha\nabla^2\mathcal{R}\right]\frac{\Delta\boldsymbol{w}^*_{\mathcal{A}}}{\Delta\theta} + X_{new,\mathcal{A}}^T\widetilde{\delta} = \mathbf{0}.$$

By taking limit $\Delta\theta \to 0$ on both sides, the above difference equation turns into ordinary differential equation system. At first, we calculate limitation on Υ_p as:

$$\lim_{\Delta\theta\to 0} \Upsilon_p = \sum_{p=3}^{\infty} \frac{(\nabla^p \mathcal{R})'}{(p-1)!} \lim_{\Delta\theta\to 0} \frac{[\boldsymbol{w}^*_{\mathcal{A}}(\theta+\Delta\theta) - \boldsymbol{w}^*_{\mathcal{A}}(\theta)]^{p-1}}{\Delta\theta},$$

which satisfies the hypotheses H_1 of L'Hospital's rule (Taylor 1952), along with the application of Proposition 1 we have:

$$\lim_{\Delta\theta\to 0} \sum_{p=3}^{\infty} \frac{(\nabla^p \mathcal{R})'}{(p-2)!} \cdot [\boldsymbol{w}^*_{\mathcal{A}}(\theta) - \boldsymbol{w}^*_{\mathcal{A}}(\theta)]^{p-2} [\boldsymbol{w}^*_{\mathcal{A}}(\theta)]'$$
$$= \sum_{p=3}^{\infty} \frac{(\nabla^p \mathcal{R})'}{(p-2)!} \cdot 0 = 0.$$

Here we have the integral ingredients to derive the ODE for exact solution path w.r.t. θ:

$$\frac{d\boldsymbol{w}^*_{\mathcal{A}}}{d\theta} = \lim_{\Delta\theta\to 0} \frac{\Delta \boldsymbol{w}^*_{\mathcal{A}}}{\Delta\theta}$$
$$= \lim_{\Delta\theta\to 0} -\left[(\theta+\Delta\theta) X^T_{new,\mathcal{A}} X_{new,\mathcal{A}} + X^T_{\mathcal{A}} X_{\mathcal{A}} + \alpha\nabla^2\mathcal{R}\right]^{-1} X^T_{new,\mathcal{A}} \widetilde{\delta},$$

which derives the conclusion of Theorem 1.

4.2 Complexity

In each iteration of Algorithm 1, we mainly solve the (4) and compute the $\mathcal{C}_g$, which has an overall computational cost of order $\mathcal{O}(|\mathcal{A}|^3)$ and $\mathcal{O}((m+\mathcal{N})n)$ (or $\mathcal{O}((m-\mathcal{N})n)$ for decremental cases).

For $m > n$, the solution of (1) could be non-unique (Roth and Fischer 2008), but the solution path can still be obtained as we solve the ODEs with initial value. When $m \gg n$ the assumption in Proposition 2 can not be held, and under this exceptional circumstances ODE takes longer time than the batch-mode mainly due to the heavier matrix computation burden on the right-hand side of (4).

5 Experiments

In this section, we first provide the experimental setup and then present our experimental results and discussion.

5.1 Experimental Setup

Our experiments are delivered from three perspectives.

Correctness. To assess the validity of our derivation at first, we employ several well-known datasets, e.g., Boston house-prices (Harrison Jr and Rubinfeld 1978), to make direct comparisons of numerical solutions $\boldsymbol{w}_0$, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, which on behalf of batch training on 5% samples (i.e., initial solution feed into algorithm), incremental training (i.e., add single sample each time) and batch training on 100% samples, respectively.

Accuracy. To prove the practicability of Algorithm 1, we compare the training process of CIGL and CDGL with batch algorithm and existing online framework for group lasso using dual averaging method, i.e., DA-GL (Yang et al. 2010). We conduct training tasks on real-world data and man-made regression datasets (MR#1). Table 2 summarizes these datasets we used. We select 25% samples in training set to perform chunk incremental updating. Moreover, we utilize the well-trained model to restore a degenerate model by chunk decremental updating for 25% samples in trained model. We compare the average Root Mean Squared Error (aRMSE) between several methods in five runs. For online DA-GL, we train it by successive addition of single data to simulate chunk updating. The α with different values are chosen to justify the effectiveness of our algorithm under various sparsity patterns.

Dataset	Size	Dimension
Yolanda	50000	100
BNG(libras_move)	30000	90
satellite_image	6435	37
BNG(wisconsin)	50000	32
cpu_act	8192	22
MR#1	100	1000
MR#2	100	2000
MR#3	100	3000

Table 2: Summary of the real-world dasetsets and synthetic data in experiments.

Efficiency. For demonstrating the performance of our algorithm, we evaluate average processing times in 10 runs when executing one chunk updating ($\mathcal{N} = 5$) using our "CIGL (CDGL)" and other methods (i.e., conventional batch-mode & existing online framework) under diverse data scales.

Additionally, we test the training time on $m > n$ cases based on MR#1∼3. In order to verify the influence of high-dimensional situation on efficiency, in our generated MR#1∼3, we keep the data size while increasing number of variable dimensions.

5.2 Implementation

We implement our Algorithm 1 in Python 3.7. Specifically, we adopt a widely-used batch training algorithm of group Lasso using FISTA optimiser (Beck and Teboulle 2009) with a gradient-based adaptive restarting scheme[1] (O'donoghue and Candes 2015). Besides, we use online learning framework for group Lasso in MATLAB code[2]. All the experiments were conducted on a Ubuntu machine with Intel 2.30GHz CPU×72 and 47.0GB RAM.

Our real-world datasets are all available online at OpenML (Vanschoren et al. 2013), we randomly selected 70% of the samples as training set for each data set. The MR#1∼3 are generated by applying a random linear regression model with no bias term in it. The standard deviation of the Gaussian centered noise applied to the output is 1. In the meantime we standardize all the datasets (i.e., standardize features by removing the mean and scaling to unit variance) using default "StandardScaler()" of Scikit-learn (Pedregosa

[1]Code available at https://github.com/yngvem/group-lasso

[2]Code available at https://hqyang.github.io/tools

Dataset	Group Partition	Regularization α	*Incremental*			*Decremental*	
			DA-GL	CIGL	Batch-mode	CDGL	Batch-mode
Yolanda	$\mathcal{P}$-15	0.2	10.066±0.229	10.027	10.027	10.230	10.230
		0.5	10.769±0.495	10.772	10.772	10.791	10.791
	$\mathcal{P}$-25	0.2	10.089±0.130	10.076	10.076	10.274	10.274
		0.5	10.849±0.133	10.791	10.791	10.791	10.791
BNG(libras_move)	$\mathcal{P}$-15	0.2	4.045±0.097	4.031	4.032	4.117	4.117
		0.4	3.966±0.233	4.287	4.287	4.341	4.341
	$\mathcal{P}$-20	0.2	4.002±0.099	4.023	4.023	4.109	4.109
		0.4	4.273±0.242	4.274	4.275	4.341	4.341
satellite_image	$\mathcal{P}$-5	0.2	1.553±0.092	1.550	1.550	1.667	1.667
		0.4	1.868±0.020	1.871	1.871	2.044	2.044
	$\mathcal{P}$-10	0.2	1.498±0.026	1.554	1.554	1.649	1.649
		0.4	1.856±0.336	1.854	1.854	2.099	2.099
BNG(wisconsin)	$\mathcal{P}$-5	0.2	29.942±0.975	29.918	29.918	29.976	29.976
		0.4	30.162±0.703	30.159	30.159	30.346	30.346
	$\mathcal{P}$-10	0.2	29.645±0.670	29.916	29.916	29.971	29.971
		0.4	30.142±0.471	30.144	30.144	30.312	30.312
cpu_act	$\mathcal{P}$-3	0.2	10.898±1.339	10.111	10.111	10.045	10.045
		0.4	11.951±0.887	10.105	10.105	10.103	10.103
	$\mathcal{P}$-5	0.2	10.339±1.038	9.987	9.987	9.915	9.915
		0.4	11.022±0.539	9.928	9.928	9.943	9.943
MR#1	$\mathcal{P}$-200	0.2	20.347±2.497	18.105	18.105	18.866	18.866
		0.4	19.351±1.388	18.042	18.042	18.896	18.897
	$\mathcal{P}$-300	0.2	20.341±2.687	18.203	18.202	18.550	18.550
		0.4	20.346±0.539	18.053	18.053	18.608	18.609

Table 3: The aRMSE (mean $\pm$ std.) results. Any variance less than 10^{-6} are omitted. "CIGL (CDGL)" and "Batch-mode" represent each iteration is trained with a chunk of new samples, or using batch algorithm to retrain from scratch, respectively. $\mathcal{P}$-i denotes each group has i features and the remaining belongs to one group.

et al. 2011), in which time consumption is negligible compared with training process.

5.3 Results & Discussions

Firstly, we place the result on Boston house in Table 4, we can intuitively see that our algorithm yields the optimal $\boldsymbol{w}$ after hundreds of iterations and recovers the right sparsity pattern. From the generalization error in Table 3, our algorithm enjoys high precision in both incremental and decremental updating, which is characterized by the nearest aRMSE to batch method compared with online framework. This indicates the effectiveness and accuracy of our algorithm.

Figure 2 and 3 present the running time. The results clearly demonstrate that there exists a huge time gap compared to other learning strategies while our CIGL & CDGL keeping the similar precision. This is due to the fact that our algorithm can track the path of $\boldsymbol{w}^*(\theta)$ without training the whole model from scratch. Although online framework can directly update analytically in a faster way, it's NOT an exact algorithm and suffers restriction in incremental learning, and more importantly, the parameters of online learning algorithms, such as γ, is ought to be tuned on the validation dataset to gain an acceptable performance, which takes researchers extra time and energy.

Furthermore, the training time in Figure 4 suggests our algorithm is inefficient in exceptional high-dimensional cases due to the increased time on solving (4). Particularly, in $m \gg n$, the advantages of CIGL & CDGL algorithm are gradually erased as m grows.

5.4 Application

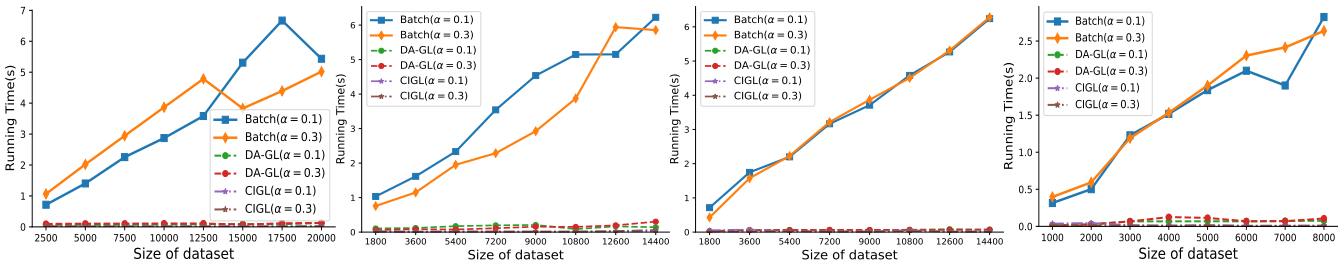
Aside from online circumstances, our proposed algorithm can also be adapted to the cases where we wish to update parameter $\boldsymbol{w}$ as the dataset changed.

Leave-one-out cross validation. For k-fold cross validation (CV), utmost k gives leave-one-out CV (LOOCV) that gives unbiased estimation of model but always costly to compute (Friedman et al. 2001). We show the histogram of the number of transition points when solving the group Lasso under LOOCV in Figure 5, similar results on more datasets are provided in Appendix C. The histograms found there exists tiny number of transition points in majority iterations, which demonstrate our incremental and decremental approach is particularly efficient given this scenario.

6 Extensions

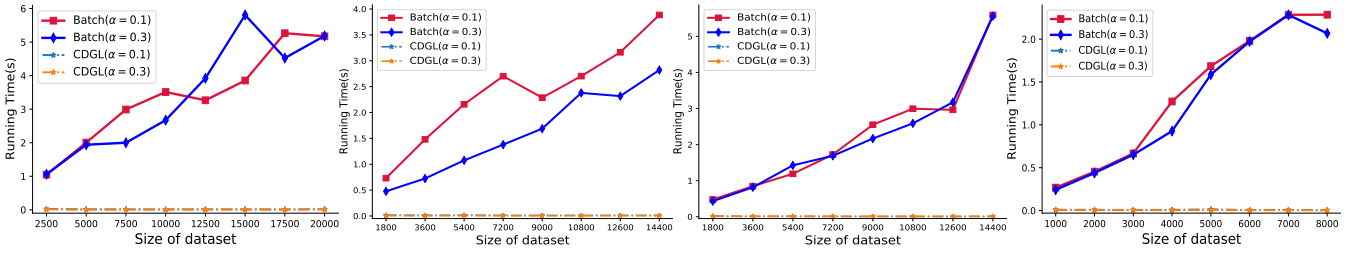
Our main idea is easily applicable to related models and we provide two heuristic conceptions here.

Extension 1 (Overlapped group Lasso) *We can extend algorithm to overlapped group Lasso (Jacob, Obozinski, and Vert 2009) while we assume that there exists at least one*



(a) Yolanda (b) BNG(libras_move) (c) BNG(wisconsin) (d) cpu_act

Figure 2: Efficiency comparison of chunk *incremental* updating.



(a) Yolanda (b) BNG(libras_move) (c) BNG(wisconsin) (d) cpu_act

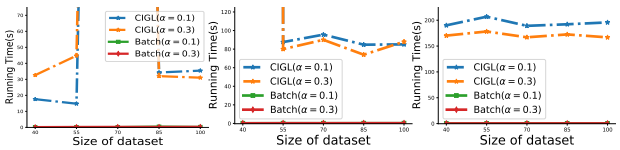
Figure 3: Efficiency comparison of chunk *decremental* updating.

Group	$\boldsymbol{w}_0$	$\boldsymbol{w}_1$	$\boldsymbol{w}_2$
I	-0.	0.	-0.
	-0.	0.	0.
	-0.	0.	-0.
II	0.	0.	0.
	-0.	0.	-0.
III	0.	1.2713	1.2712
	0.	-0.4367	-0.4367
	-0.	0.0217	0.0217
	-0.	-0.2584	-0.2584
	-0.	-0.5755	-0.5756
IV	0.	-1.3899	-1.3898
	0.	0.7064	0.7065
	-0.	-2.2596	-2.2596

Table 4: Solutions comparison on Boston house, where $\boldsymbol{w}_0$ is initial solution on 5% samples, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ represent solutions on 100% samples using our algorithm or batch training, respectively.

feature belonging to distinct groups. In this more general situation, we can simply duplicate the overlapped variables as performed similarly in (Jacob, Obozinski, and Vert 2009), then the input data $X \in \mathbb{R}^{m\times n}$ *turns into* $\widehat{X} \in \mathbb{R}^{m\times\sum|g|}(g = 1, .., G)$. *Thus Algorithm 1 can handle it to compute the solution in overlapped case.*

Extension 2 (Sparse group Lasso) *Setting the* $\mathcal{R}_\Omega(\boldsymbol{w})$ *in equation (2) as* $\alpha\sum_{g=1}^{G}\sqrt{d_g}\,\|\boldsymbol{w}_g\|_2 + \alpha_2\|\boldsymbol{w}\|_1$ *gives the sparse group Lasso regularized model (Simon et al. 2013). We found out that the ODE (4) is also applicable to this problem. We provide experimental verification in Appendix D. Meanwhile (3) should be recast as* $\mathcal{C}_g = \alpha\sqrt{d_g}\frac{\boldsymbol{w}_g}{\|\boldsymbol{w}_g\|} + \alpha_2\boldsymbol{u}_g$, *where* $\boldsymbol{u} \in \partial\|\boldsymbol{w}\|_1 = \begin{cases} [-1,1] & \boldsymbol{w}_i = 0 \\ sign(\boldsymbol{w}_i) & else \end{cases}$. *Thus the threshold condition not only includes group-wise sparsity but also zero coefficients within each active group (i.e. within group sparsity). Our idea can apply but a more effective manner for detecting transition point still need to be explored in future.*



(a) MR#1 (b) MR#2 (c) MR#3

Figure 4: Running time (in seconds) on MR#1∼3.

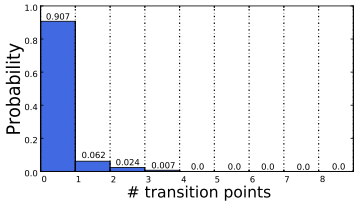


Figure 5: Histogram of LOOCV. The dataset with size 1000×100 is generated similar to MR#1∼3.

7 Conclusion

To alleviate group Lasso's extensive computation cost in dynamic scenario, the CIGL & CDGL is thus proposed. The experimental results confirm that our method is more efficient than the existing online group Lasso and batch algorithm while retaining high accuracy.

References

Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A. D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. 2010. A view of cloud computing. *Communications of the ACM*, 53(4): 50–58.

Bach, F. R. 2008. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9(6).

Beck, A.; and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1): 183–202.

Bottou, L.; and LeCun, Y. 2004. Large scale online learning. *Advances in neural information processing systems*, 16: 217–224.

Boyd, S.; Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Casella, G.; Ghosh, M.; Gill, J.; and Kyung, M. 2010. Penalized regression, standard errors, and Bayesian lassos. *Bayesian analysis*, 5(2): 369–411.

Chatterjee, S.; Steinhaeuser, K.; Banerjee, A.; Chatterjee, S.; and Ganguly, A. 2012. Sparse group lasso: Consistency and climate applications. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, 47–58. SIAM.

Duchi, J. C.; and Singer, Y. 2009. Efficient Learning using Forward-Backward Splitting. In *NIPS*, volume 22, 495–503.

Friedman, J.; Hastie, T.; Tibshirani, R.; et al. 2001. *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Garrigues, P.; and Ghaoui, L. 2008. An homotopy algorithm for the Lasso with online observations. *Advances in neural information processing systems*, 21: 489–496.

Gepperth, A.; and Hammer, B. 2016. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (ESANN)*.

Giraud-Carrier, C. 2000. A note on the utility of incremental learning. *Ai Communications*, 13(4): 215–223.

Golub, G. H.; and Van Loan, C. F. 2013. *Matrix computations*, volume 3. JHU press.

Harrison Jr, D.; and Rubinfeld, D. L. 1978. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1): 81–102.

Huo, Y.; Xin, L.; Kang, C.; Wang, M.; Ma, Q.; and Yu, B. 2020. SGL-SVM: a novel method for tumor classification via support vector machine with sparse group Lasso. *Journal of Theoretical Biology*, 486: 110098.

Ida, Y.; Fujiwara, Y.; and Kashima, H. 2019. Fast sparse group lasso.

Jacob, L.; Obozinski, G.; and Vert, J.-P. 2009. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, 433–440.

Li, Z.; Li, Y.; Feng, W.; Fei, Y.; and Zheng-Long, X. 2014. Efficient and accelerated online learning for sparse group LASSO. In *2014 IEEE International Conference on Data Mining Workshop*, 1171–1177. IEEE.

Lim, M.; and Hastie, T. 2015. Learning interactions via hierarchical group-lasso regularization. *Journal of Computational and Graphical Statistics*, 24(3): 627–654.

Ma, S.; Song, X.; and Huang, J. 2007. Supervised group Lasso with applications to microarray data analysis. *BMC bioinformatics*, 8(1): 1–17.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Meier, L.; Van De Geer, S.; and Bühlmann, P. 2008. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1): 53–71.

O'donoghue, B.; and Candes, E. 2015. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3): 715–732.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Rao, N.; Nowak, R.; Cox, C.; and Rogers, T. 2015. Classification with the sparse group lasso. *IEEE Transactions on Signal Processing*, 64(2): 448–463.

Rosset, S.; and Zhu, J. 2007. Piecewise linear regularized solution paths. *The Annals of Statistics*, 1012–1030.

Roth, V.; and Fischer, B. 2008. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th international conference on Machine learning*, 848–855.

Simon, N.; Friedman, J.; Hastie, T.; and Tibshirani, R. 2013. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2): 231–245.

Simon, N.; and Tibshirani, R. 2012. Standardization and the group lasso penalty. *Statistica Sinica*, 22(3): 983.

Taylor, A. E. 1952. L'Hospital's rule. *The American Mathematical Monthly*, 59(1): 20–24.

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

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2): 49–60.

Xiao, L. 2010. Dual averaging methods for regularized stochastic learning and online optimization.

Yang, H.; Xu, Z.; King, I.; and Lyu, M. R. 2010. Online learning for group lasso. In *ICML*.

Yau, C. Y.; and Hui, T. S. 2017. LARS-type algorithm for group lasso. *Statistics and Computing*, 27(4): 1041–1048.

Yuan, M.; and Lin, Y. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1): 49–67.

Zhang, H.; Wang, J.; Sun, Z.; Zurada, J. M.; and Pal, N. R. 2019. Feature selection for neural networks using group lasso regularization. *IEEE Transactions on Knowledge and Data Engineering*, 32(4): 659–673.

Zhang, Y.; Zhang, N.; Sun, D.; and Toh, K.-C. 2020. An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems. *Mathematical Programming*, 179(1): 223–263.

Zhou, H.; and Wu, Y. 2014. A generic path algorithm for regularized statistical estimation. *Journal of the American Statistical Association*, 109(506): 686–699.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, 928–936.