When Online Learning Meets ODE: Learning without Forgetting on Variable Feature Space

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

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

Abstract

Machine learning systems that built upon varying feature space are ubiquitous across the world. When the set of practical or virtual features changes, the online learning approach can adjust the learned model accordingly rather than re-training from scratch and has been an attractive area of research. Despite its importance, most studies for algorithms that are capable of handling online features have no ensurance of stationarity point convergence, while the accuracy guaranteed methods are still limited to some simple cases such as ℓ_1 or ℓ_2 norms with square loss. To address this challenging problem, we develop an efficient Dynamic Feature Learning System (DFLS) to perform online learning on the unfixed feature set for more general statistical models and demonstrate how DFLS opens up many new applications. We are the first to achieve accurate & reliable feature-wise online learning for a broad class of models like logistic regression, spline interpolation, group Lasso and Poisson regression. By utilizing DFLS, the updated model is theoretically the same as the model trained from scratch using the entire new feature space. Specifically, we reparameterize the feature-varying procedure and devise the corresponding ordinary differential equation (ODE) system to compute the optimal solutions of the new model status. Simulation studies reveal that the proposed DFLS can substantially ease the computational cost without forgetting.

1 Introduction

On a wide range of real-world machine learning (ML) applications, the attributes / features of input may dynamically change and it is often computationally unrealistic to train over the entire dataset (Agarwal, Saradhi, and Karnick 2008; Fontenla-Romero et al. 2013). Practically, in bioinformatics, acquiring the entire set of features for every training instance is expensive due to the pricey cost of conducting wet-lab experiments (Wang et al. 2013a). In real-time sensing, inactive sensors or obsolete historical attributes are demanded to be dropped from the existing feature space (You et al. 2018), or new attributes should be included since new sensors are put into work. For Internet applications like spam filtering or personalized news feeds, textual data are usually represented as a bag-of-words and the length of the feature vector is dynamically changing (Giraud-Carrier 2000; Dekel, Shamir, and Xiao 2010).

Alongside these motivations, the virtual (or generated / transformed) feature space is also not static. In polynomial regression (Max and Burkhart 1976), we need to add / delete features of the generated polynomial features when altering the degree or interaction terms. Likewise, by the time of using random Fourier features (Rahimi and Recht 2007) in kernel machines, we may want to tune the approximation level of the kernel space. Similar adjustments exist in deep representation learning for expandable networks (Yoon et al. 2018; Parisi et al. 2019). The above examples describe scenarios where new features can be introduced or some of the existing features can cease to exist. Additional concrete scenarios are offered in Appendix C.1. The Figure 1 shows a schematic diagram of our problem setting.

Traditional ML is performed offline and a learning algorithm can go over the full batch of data multiple times (Beyazit 2021). Rather than attempt to fully re-training from scratch using batch algorithms, the online learning (Bottou and Cun 2003; Fontenla-Romero et al. 2013) is now generally conceded to offer a better approach to solve a model in an online setup. This leads to a model which is able to readily extrapolate beyond the current input space, thereby has become an active research field in ML communities (Wu et al. 2019). Whilst a number of previous works have achieved the feature-wise online learning to a certain extent, they are either developed for specific statistical problems such as primitive ℓ_1 and ℓ_2 regularization with square loss (Freund, Grigas, and Mazumder 2013; Hoerl and Kennard 1970), or turn out to be approximate methods, in which their convergence to the stationarity point on the new feature set cannot be guaranteed (Beyazit, Alagurajah, and Wu 2019; He et al. 2021). In this paper, we aim to develop an *accuracy guaranteed* Dynamic Feature Learning System (DFLS) to perform training on the unfixed feature set for a broad class of statistical models, with the motivation behind this approach being that many real-world ML systems are running with non-static feature spaces. By utilizing DFLS, the updated model *is exactly the same as the model trained from scratch* using the entire new feature space. *i.e.*, the model can refine its old knowledge (without suffering from forgetting) when feature space varies.

Training an optimal model on the unfixed feature set is notoriously difficult to solve since it suffers from the following

*Corresponding Author

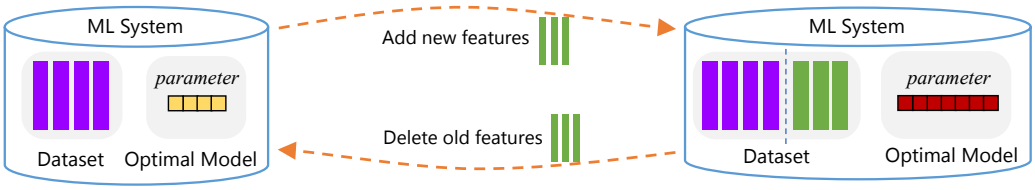


Figure 1: Diagram of online learning on variable feature space. A nature idea is to train from scratch on the changed dataset. When using DFLS, the new optimal model can be obtained rather quickly without retraining as the feature space changed.

limitations. First, the dimensionality of model variable is mutable when the input feature space makes changes. The vast majority of previous efforts in online learning failed to overcome this scenario. Second, we study the problem of a more general linear model, which owns a more complicated structure rather than heavily investigated square loss or primitive ℓ_1, ℓ_2 penalty. This setting brings difficulties to the theoretical analysis, due to the fact that first-order necessary condition is far more complex. This poses a real challenge. Consequently, how to perform accurate feature-varying online learning for a great many of well-known models (*e.g.*, ridge logistic estimator (Schaefer, Roi, and Wolfe 1984), group Lasso (Yuan and Lin 2006), Poisson regression (Frome 1983)) remain vexing issues.

To work around these bottlenecks, a plausible idea is to conduct serialization between the two discrete statuses of models (*i.e.*, optimal models that fit with two different feature spaces). We first draw insights from recursive Lasso (Hofleitner et al. 2014), so as to reparameterize the feature-varying procedure. Based on the auxiliary variable and the first-order necessary condition, we are able to build the system of ordinary differential equations (ODE), which is the bridge connecting two different model statuses. In an effort to pursue a faster computational speed, we also propose several (optional) advanced optimization techniques for DFLS. Extensive experimental results validate its efficiency and accuracy compared to the existing batch training algorithms.

Our Contributions The main contributions of this paper are summarized as follows.

1. We develop a *provable accurate* learning framework DFLS to perform online learning on the unfixed feature set with ODE and sets control. On the new feature set, it is theoretically found that the convergence of stationarity point *w.r.t.* cost function can be guaranteed.
2. We demonstrate how DFLS opens up many new applications for a broad class of well-known linear models like logistic regression, spline interpolation, group Lasso and Poisson regression with varying features.
3. Several advanced acceleration techniques are designed for our DFLS and we successfully expand our DFLS to fit the nonlinear patterns in some manners.

Notations $\nabla\mathcal{R}(x)$ is the gradient of $\mathcal{R}(\cdot)$ at the point x, $\nabla^2\mathcal{R}(x)$ denotes a Hessian matrix. $\mathcal{I}$ represents identity matrix. The $diag(B_i)$ denotes diagonal matrix. $\mathbf{X}_{\mathcal{A}}$ means the columns of $\mathbf{X}$ are indexed by the indices set $\mathcal{A}$. $\mathbf{X}_i$ denotes i-th row of $\mathbf{X}$. The sigmoid function $\sigma(\mathrm{x}) = (1 + e^{-\mathrm{x}})^{-1}$ and $\odot$ denotes the Hadamard product (*i.e.*, $C = A \odot B \Rightarrow C_{i,j} = A_{i,j} * B_{i,j}$). Given a $\xi \in \mathbb{R}^{s\times 1}$, we define the spread operator $\mathrm{SpO}_t(\xi) \overset{\text{def}}{=} [\ \xi \quad \xi \quad \cdots \quad \xi\ ] \in \mathbb{R}^{s\times t}$.

Reproducibility Python codes for reproducing the key results are available in the supplementary materials.

2 Related Work

The quest for algorithms that handle online streaming features has never ceased in recent years. Authors in (Wu et al. 2010, 2012; Wang et al. 2013a, 2015; Li et al. 2013; Wang et al. 2013b; Hu et al. 2018b; Zhou et al. 2020) derived a range of online (group) feature selection algorithms, which focus on the problem of feature selection in an online fashion, rather than training an optimal model within new feature space. The (Hou and Zhou 2017) designs an approach for one-pass learning with incremental and decremental features. By utilizing a diversity generation strategy and feature incremental tree, (Hu et al. 2018a) founded feature-wise online learning for random forest. The model developed by (Beyazit, Alagurajah, and Wu 2019) considers binary linear classifiers for varying feature spaces. The (Beyazit 2021; He et al. 2021) are designed for tackling the problem of online semi-supervised learning. The (He et al. 2019, 2020) propose a generative graphical model to fit capricious data streams. However, none of them can guarantee the theoretical precision of model, due to the fact that the updated model is *non-equivalent* to the model re-trained on the whole new feature space.

In the field of accuracy guaranteed online learning, the Lasso model has been solved in (Freund, Grigas, and Mazumder 2013) using the forward stagewise regression. Other simple regularization like ℓ_2-norm leads to analytical solutions (Hoerl and Kennard 1970), hence result can be computed directly in the new feature space. To date, the online learning algorithm with first-order optimal solution guarantee is still awaiting to be developed for a wide range of models since existing approaches are not scalable and can only solve specific learning problems. Similar to the feature-wise online scenario studied here, the (Zhang et al. 2015, 2016; Alagurajah, Yuan, and Wu 2020) surveyed online learning from trapezoidal data streams and (Hou, Zhang, and Zhou 2017; Zhang et al. 2020; Hou et al. 2021) investigated the feature evolvable streams.

Comparison with Homotopy Method While there are likely to have some mathematical similarities, our DFLS differs from the family of homotopy algorithms (Watson 1986;

Nocedal and Wright 1999; Watson 2001). As mentioned previously, the work of (Garrigues and Ghaoui 2008) (for Lasso) and (Hofleitner et al. 2014, 2013; Li and Gu 2022) (for generalized Lasso) also incorporate instrumental variable(s) in order to realize the online learning, however, the authors achieve this by a series of linear systems, rather than specifying an ODE-based structure as in our approach. More to the point, they are all focused on the **sample-wise** online learning and limited to mere square loss. In contrast, DFLS can solve a *more general* problem by the technique of ODEs, which is far beyond existing homotopy algorithms.

3 Setting

Given the dataset $\mathcal{D} = \{\mathbf{X} \in \mathbb{R}^{n\times d}, \mathbf{y} \in \mathbb{R}^n\}$ with n observations, each observation $\mathbf{X}_i \in \mathbb{R}^d$ has d features. The $\mathbf{y}_i \in \mathbb{R}$ for the regression problem and $\mathbf{y}_i \in \{0, 1\}$ for the binary classification task. In supervised learning, an extensive number of algorithms essentially solve the regularized minimization problem, and for a linear model, it reads

$$\min_{\beta\in\mathbb{R}^d} \lambda R(\beta) + L(\mathbf{X}\beta, \mathbf{y}), \tag{1}$$

where λ is the regularization parameter to balance the prediction loss and penalty. The $R(\cdot)$ is the regularization term and $L(\cdot)$ denotes the loss function. The β represents learnable parameters inside the model.[1]

Assumption 1. *We assume that the loss function $L(\cdot)$ and regularization item $R(\cdot)$ in (1) are piecewise (or elementwise) second-order differentiable.*

Assumption 2. *We assume that the optimal solution of (1) can be achieved.*

First, we consider the case of adding new features shown in Figure 1. Assume we currently own n labeled instances $\mathcal{D}^{cur} = \{\mathbf{X}, \mathbf{y}\}$ and an optimal ML model trained using $\mathcal{D}^{cur}$, in which the parameter is denoted by $\beta^{(1)} \in \mathbb{R}^d$ (already known). With the time passed, newly acquired data can be formalized as $\widetilde{\mathbf{X}} \in \mathbb{R}^{n\times d'}$ that includes d' features and we want to add $\widetilde{\mathbf{X}}$ into the existing training set.[2] In other words, we need an optimal ML model that trained on $\mathcal{D}^{new} = \left\{\left[\mathbf{X}, \widetilde{\mathbf{X}}\right], \mathbf{y}\right\}$ and the parameter of optimal model on the new dataset is $\beta^{(2)} \in \mathbb{R}^{d+d'}$ (currently unknown). A conventional practice is to re-train a new model on $\mathcal{D}^{new}$, while our DFLS aims to compute the $\beta^{(2)}$ directly (*i.e.*, without re-training from scratch) using the information of $\beta^{(1)}$.

Conversely, when we think about deleting d' old features $\widetilde{\mathbf{X}} \in \mathbb{R}^{n\times d'}$ from existing dataset $\mathcal{D}^{cur} = \left\{\left[\mathbf{X}, \widetilde{\mathbf{X}}\right], \mathbf{y}\right\}$, our goal is to restore an optimal ML model trained on $\mathcal{D}^{new} = \{\mathbf{X}, \mathbf{y}\}$ in like manner.

[1] The bias term in target can be introduced by expanding $\mathbf{X}$ with one column whose all elements are 1.

[2] To simplify notations we assume the new $\widetilde{\mathbf{X}}$ appears last.

4 Methodology

4.1 Reparameterization

For uniformity, we organize the training data as $\left[\mathbf{X}, \widetilde{\mathbf{X}}\right]$ and corresponding coefficient $\beta = \begin{bmatrix} \beta_1 \in \mathbb{R}^d \\ \beta_2 \in \mathbb{R}^{d'} \end{bmatrix}$ by two types of features, where $\widetilde{\mathbf{X}}$ contains new (or obsolete) features that we intend to add into (or delete from) the existing feature set.

Definition 1 (Help Function φ_θ)**.** *The following two conditions hold for a help function $\varphi_\theta \in \mathbb{R}$, where the argument θ is a scalar.* ***(a.)*** *The inverse function of φ_θ (denotes as φ_θ^{-1}) exists and should be continuous.* ***(b.)*** *The first-order derivative of φ_θ w.r.t. θ is smooth in closed interval $\left[\underline{\theta},\ \bar{\theta}\ \right]$, where $\underline{\theta} = \varphi_\theta^{-1}(0)$ and $\bar{\theta} = \varphi_\theta^{-1}(1)$.*

We work with the following formulation with help function augmentation as

$$\min_{\beta\in\mathbb{R}^{d+d'}} \lambda R(\beta) + L\left(\left[\mathbf{X}, \varphi_\theta\widetilde{\mathbf{X}}\right]\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}, \mathbf{y}\right). \tag{2}$$

A rational help function φ_θ in (2) can be used to reparameterize the addition or subtraction procedure of a chunk feature $\widetilde{\mathbf{X}}$. In other words, the $\varphi_\theta = 0$ in (2) consists with the optimal model trained on d features, while $\varphi_\theta = 1$ corresponds with the optimal model trained with $d + d'$ features, which build the bridge connecting two different model statuses and again indicate the first-order optimal solution can be achieved by DFLS.

Assumption 3. *The optimal solution $\left\{\beta(\theta)|\theta \in \left[\underline{\theta},\ \bar{\theta}\ \right]\right\}$ is a continuous function w.r.t. θ.*

The Assumption 3 is the foundation of constructing an ODE system to perform feature-varying online learning. Here we give the hypothesis of mentioned continuity, and the formal theorem will be presented till Sec. 6 for the specific learning model since its proof involves concrete optimization objectives. To accomplish feature-adding online training, we are supposed to calculate the solution of $\beta(\theta)$ in (2) at $\theta = \bar{\theta}$. Conversely, the solution of model that eliminating features in $\widetilde{\mathbf{X}}$ can be computed at $\theta = \underline{\theta}$.

4.2 First-order ODE System

In this part we provide a unified first-order ODE framework for online learning. To represent the sparsity enforcing penalties and non-sparse models together, we denote the $\mathcal{A}$ and $\mathcal{E}$ are active sets that containing indices of β_1 and β_2 that $\beta_{ij} \neq 0$ $(i = 1, 2)$, so as to store active components, *i.e.*, we have $\beta_1 = \left(\beta_{\mathcal{A}}^T, \beta_{\overline{\mathcal{A}}}^T\right)^T$, $\beta_2 = \left(\beta_{\mathcal{E}}^T, \beta_{\overline{\mathcal{E}}}^T\right)^T$, where $\beta_{\overline{\mathcal{A}}}, \beta_{\overline{\mathcal{E}}}$ remains $\mathbf{0}$.

Theorem 1. *When the active sets $\mathcal{A}$ and $\mathcal{E}$ are fixed, the optimal β of (2) satisfies the following ODE system*

$$\widehat{\mathcal{K}}\begin{bmatrix} \frac{d\beta_{\mathcal{A}}}{d\theta} \\ \frac{d\beta_{\mathcal{E}}}{d\theta} \end{bmatrix} = -\begin{bmatrix} \frac{\partial^2 L(\cdot)}{\partial\beta_{\mathcal{A}}\partial\theta} + \lambda\frac{\partial^2 R(\cdot)}{\partial\beta_{\mathcal{A}}\partial\theta} \\ \frac{\partial^2 L(\cdot)}{\partial\beta_{\mathcal{E}}\partial\theta} + \lambda\frac{\partial^2 R(\cdot)}{\partial\beta_{\mathcal{E}}\partial\theta} \end{bmatrix}. \tag{3}$$

The key matrix $\widehat{\mathcal{K}}$ has the form of $\begin{bmatrix} U & Z^T \\ Z & V \end{bmatrix}$, *where* $U = \frac{\partial^2 L(\cdot)}{\partial \beta_{\mathcal{A}}^2} + \lambda \frac{\partial^2 R(\cdot)}{\partial \beta_{\mathcal{A}}^2}$, $V = \frac{\partial^2 L(\cdot)}{\partial \beta_{\mathcal{E}}^2} + \lambda \frac{\partial^2 R(\cdot)}{\partial \beta_{\mathcal{E}}^2}$ *and* $Z = \frac{\partial^2 L(\cdot)}{\partial \beta_{\mathcal{A}} \partial \beta_{\mathcal{E}}} + \lambda \frac{\partial^2 R(\cdot)}{\partial \beta_{\mathcal{A}} \partial \beta_{\mathcal{E}}}$.

Remark 1. *The $\beta_{\mathcal{A}}$ (or $\beta_{\mathcal{E}}$) will revert to default β_1 (or β_2) when non-sparsity $R(\cdot)$ is applied. Our Table 1 summarizes some results of derived ODE systems using Theorem 1.*

Note that all of the proofs of main body are listed in Appendix A. Throughout, the problem (3) is solved exactly using the numerical solver of ODE, in which the direct matrix inverse of $\widehat{\mathcal{K}}$ is required. Further, when applying `DFLS` to some practical scenarios, it may turn out that $\widehat{\mathcal{K}}$ is symmetric or positive definite when conducting theoretical expressions of $\widehat{\mathcal{K}}$. Under the circumstances, we can utilize Bunch-Kaufman factorization or Cholesky decomposition on key matrix $\widehat{\mathcal{K}}$ to raise the efficiency (Golub and Van Loan 2013), which essentially solves (3) with nearly $\mathcal{O}(|\mathcal{A} \cup \mathcal{E}|^3)$ flops. The `DFLS` operates by numerically solving an initial-value differential equation on $\left[\underline{\theta},\ \bar{\theta}\right]$ segment by segment, where the solution β regarding to θ can be computed swiftly before active sets $\mathcal{A}$ or $\mathcal{E}$ changes.

Segmentation The segmentation is owing to indices change in $\mathcal{A}$ or $\mathcal{E}$, which is characterized by the variation of optimality conditions. Meanwhile the ODE wouldn't work correctly in a new segment due to $\mathcal{A}$ or $\mathcal{E}$ are outdated. We can keep observation on the optimality conditions while solving the β to estimate whether there enters a new segmentation, and update the active sets $\mathcal{A}$ and $\mathcal{E}$ accordingly. The intuitions of Theorem 1, as well as the segmentation, are given in Appendix C.2.

Remark 2. *When non-sparsity regularization $R(\cdot)$ is used, `DFLS` needn't cost extra time on detecting and refreshing the sets $\mathcal{A}$, $\mathcal{E}$, which apparently reduces the computation complexity.*

Computational Complexity Detecting active sets $\mathcal{A}$ and $\mathcal{E}$ by their partition thresholding takes $\mathcal{O}(|\mathcal{A} \cup \mathcal{E}|)$ flops. In addition, the cost of $\mathcal{O}(|\mathcal{A} \cup \mathcal{E}|^3)$ per evaluation as mentioned before is not as daunting as it appears. Assume that the active set $\mathcal{A}$ and $\mathcal{E}$ are known, even the fastest convergence rate second-order Newton's method (Ehrlich 1967) needs to solve the similar evaluation multiple times until convergence. The efficiency of `DFLS` lies in the fact that no iterations are needed at any θ and the solver adaptively chooses step sizes to solve the first-order ODE system on $\left[\underline{\theta},\ \bar{\theta}\right]$.

5 Advanced `DFLS`

The following operations are *optional* in implementation to further improve or accelerate the basic `DFLS`.

5.1 Interval Pruning

Remark 3. *The pruning stage can be performed only when sparsity-inducing penalties are used.*

As mentioned earlier, `DFLS` indeed solves an ODE system segment by segment. For sparse models, due to the nature of subgradient, the original interval $\left[\underline{\theta},\ \bar{\theta}\right]$ can thus be pruned to make the solver skip the first segment. We calculate the end of the first segment $\widetilde{\theta}$ in exact solution as

$$\varphi_\theta|_{\theta=\widetilde{\theta}} = \min\left\{ \frac{\lambda \|\nabla_{\beta_i} \mathcal{R}(\beta)\|}{\|\nabla_{\beta_i} L(\mathbf{X}\beta, \mathbf{y})\|} \right\},\ \text{for } i \in \overline{\mathcal{E}}, \quad (4)$$

then we update $\beta|_{\theta=\widehat{\theta}} \leftarrow \beta|_{\theta=\underline{\theta}}$, where $\widehat{\theta} = \min\left\{\widetilde{\theta}, \bar{\theta}\right\}$.

The analysis of (4) is in Appendix A. If clipped $\widehat{\theta}$ is exactly $\varphi_\theta^{-1}(1)$, then we can finish executing the algorithm. From expression in (4) and Figure 2, we know that the length of pruned segment is piecewise proportional to hyperparameter λ, which makes our approach very efficient when λ is in relatively large quantities.

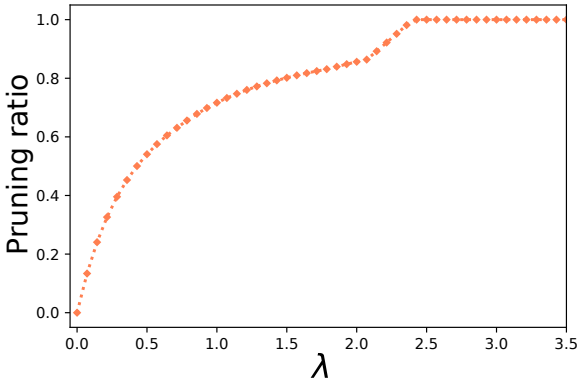


Figure 2: Pruning proportion of interval $\left[\underline{\theta},\ \bar{\theta}\right]$ w.r.t. λ on Boston house dataset (Harrison Jr and Rubinfeld 1978). The ratio is computed via $\frac{\widetilde{\theta}-\underline{\theta}}{\bar{\theta}-\underline{\theta}}$. More results are in Appendix F.3.

Remark 4. *Note that when we deleting features, there are no analytical solution for $\widehat{\theta}$. The condition of early stopping is that whether $\mathcal{E} = \varnothing$ (or $\beta_2 = \mathbf{0}$).*

5.2 Optimizing the Query Complexity

Any ODE solver repeatedly queries the derivative of β. We first point out that the query complexity of $\mathcal{O}(|\mathcal{A} \cup \mathcal{E}|^3)$ may be unbearable in extra high-dimensional data. In this setting our `DFLS` involves sweep operator (Goodnight 1979; Lange, Chambers, and Eddy 1999; Little and Rubin 2019). Suppose $\boldsymbol{A}$ is an $m \times m$ symmetric matrix, sweeping on the k-th diagonal entry $a_{kk} \neq 0$ of $\boldsymbol{A}$ yields a matrix $\widehat{\boldsymbol{A}}$ with entries

$$\hat{a}_{kk} = -\frac{1}{a_{kk}} \qquad \hat{a}_{ik} = \frac{a_{ik}}{a_{kk}},\ i \neq k$$
$$\hat{a}_{kj} = \frac{a_{kj}}{a_{kk}},\ j \neq k \qquad \hat{a}_{ij} = a_{ij} - \frac{a_{ik}a_{kj}}{a_{kk}},\ i, j \neq k.$$

Because sweep operator preserve symmetry, all operations can be carried out on either the lower or upper-triangular part of $\boldsymbol{A}$ alone and ease the computational burden.

Definition 2. *Given a block matrix* $\boldsymbol{A} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, *if D is invertible, then the Schur complement (Zhang 2006) of the block D of the matrix $\boldsymbol{A}$ is defined by $\boldsymbol{A}/A := D - CA^{-1}B$.*

Type of Loss	First-order ODE System
Square Loss (Sec. 6.1)	$\left[\begin{array}{c\|c} \frac{1}{n}\mathbf{X}_{\mathcal{A}}^{\mathbf{T}}\mathbf{X}_{\mathcal{A}}+\alpha\nabla^{\mathbf{2}}_{\beta_{\mathcal{A}}}\mathcal{R} & * \\ \frac{\varphi_\theta}{n}\cdot\widetilde{\mathbf{X}}_{\mathcal{E}}^{T}\mathbf{X}_{\mathcal{A}} & \frac{\varphi_\theta^2}{n}\widetilde{\mathbf{X}}_{\mathcal{E}}^{T}\widetilde{\mathbf{X}}_{\mathcal{E}}+\alpha\nabla^2_{\beta_{\mathcal{E}}}\mathcal{R} \end{array}\right]\left[\begin{array}{c}\frac{d\beta_{\mathcal{A}}}{d\theta}\\ \frac{d\beta_{\mathcal{E}}}{d\theta}\end{array}\right] = -\frac{\varphi'_\theta}{n}\left[\begin{array}{c}\mathbf{X}_{\mathcal{A}}^{\mathbf{T}}\widetilde{\mathbf{X}}_{\mathcal{E}}\beta_{\mathcal{E}}\\ \widetilde{\mathbf{X}}_{\mathcal{E}}^{T}\left(\widehat{\boldsymbol{e}(\theta)}+\varphi_\theta\widetilde{\mathbf{X}}_{\mathcal{E}}\beta_{\mathcal{E}}\right)\end{array}\right]$
Cross Entropy Loss (Sec. 6.2)	$\left[\begin{array}{c\|c} CX^T\left[\mathrm{SpO}_d(\mathcal{S}(\boldsymbol{\upsilon}))\odot\mathbf{X}\right]+\mathcal{I} & * \\ C\varphi_\theta\cdot\widetilde{\mathbf{X}}^T\left[\mathrm{SpO}_d(\mathcal{S}(\boldsymbol{\upsilon}))\odot\mathbf{X}\right] & C\varphi_\theta^2\widetilde{\mathbf{X}}^T\left[\mathrm{SpO}_{d'}(\mathcal{S}(\boldsymbol{\upsilon}))\odot\widetilde{\mathbf{X}}\right]+\mathcal{I}\end{array}\right]\left[\begin{array}{c}\frac{d\beta_1}{d\theta}\\ \frac{d\beta_2}{d\theta}\end{array}\right] = -C\varphi'_\theta\left[\begin{array}{c}\mathbf{X^T}\left[\mathcal{S}(\boldsymbol{\upsilon})\odot(\widetilde{\mathbf{X}}\boldsymbol{\beta_2})\right]\\ \widetilde{\mathbf{X}}^T\left(\sigma(\boldsymbol{\upsilon})-\mathbf{y}\right)+\varphi_\theta\widetilde{\mathbf{X}}^T\left[\mathcal{S}(\boldsymbol{\upsilon})\odot(\widetilde{\mathbf{X}}\beta_2)\right]\end{array}\right]$
Poisson Deviance (Sec. 6.3)	$\left[\begin{array}{c\|c} \frac{1}{n}\mathbf{X}^T\left[\mathrm{SpO}_d(e^{\boldsymbol{\upsilon}})\odot\mathbf{X}\right]+2\lambda\mathcal{I} & * \\ \frac{\varphi_\theta}{n}\cdot\widetilde{\mathbf{X}}^T\left[\mathrm{SpO}_d(e^{\boldsymbol{\upsilon}})\odot\mathbf{X}\right] & \frac{\varphi_\theta^2}{n}\widetilde{\mathbf{X}}^T\left[\mathrm{SpO}_{d'}(e^{\boldsymbol{\upsilon}})\odot\widetilde{\mathbf{X}}\right]+2\lambda\mathcal{I}\end{array}\right]\left[\begin{array}{c}\frac{d\beta_1}{d\theta}\\ \frac{d\beta_2}{d\theta}\end{array}\right] = -\frac{\varphi'_\theta}{n}\left[\begin{array}{c}\mathbf{X}^T\left[e^{\boldsymbol{\upsilon}}\odot(\widetilde{\mathbf{X}}\beta_2)\right]\\ \widetilde{\mathbf{X}}^T\left(e^{\boldsymbol{\upsilon}}-\boldsymbol{y}\right)+\varphi_\theta\widetilde{\mathbf{X}}^T\left[e^{\boldsymbol{\upsilon}}\odot(\widetilde{\mathbf{X}}\beta_2)\right]\end{array}\right]$
Gamma Deviance (Sec. 6.3)	$\left[\begin{array}{c\|c} \frac{1}{n}\mathbf{X}^T\left[\mathrm{SpO}_d(\frac{\mathbf{y}}{e^{\boldsymbol{\upsilon}}})\odot\mathbf{X}\right]+2\lambda\mathcal{I} & * \\ \frac{\varphi_\theta}{n}\cdot\widetilde{\mathbf{X}}^T\left[\mathrm{SpO}_d(\frac{\mathbf{y}}{e^{\boldsymbol{\upsilon}}})\odot\mathbf{X}\right] & \frac{\varphi_\theta^2}{n}\widetilde{\mathbf{X}}^T\left[\mathrm{SpO}_{d'}(\frac{\mathbf{y}}{e^{\boldsymbol{\upsilon}}})\odot\widetilde{\mathbf{X}}\right]+2\lambda\mathcal{I}\end{array}\right]\left[\begin{array}{c}\frac{d\beta_1}{d\theta}\\ \frac{d\beta_2}{d\theta}\end{array}\right] = -\frac{\varphi'_\theta}{n}\left[\begin{array}{c}\mathbf{X}^T\left[\frac{\mathbf{y}}{e^{\boldsymbol{\upsilon}}}\odot(\widetilde{\mathbf{X}}\beta_2)\right]\\ \widetilde{\mathbf{X}}^T\left(1-\frac{\mathbf{y}}{e^{\boldsymbol{\upsilon}}}\right)+\varphi_\theta\widetilde{\mathbf{X}}^T\left[\frac{\mathbf{y}}{e^{\boldsymbol{\upsilon}}}\odot(\widetilde{\mathbf{X}}\beta_2)\right]\end{array}\right]$

$\boldsymbol{\upsilon}=\mathbf{X}\beta_1+\varphi_\theta\widetilde{\mathbf{X}}\beta_2$, $\widehat{\boldsymbol{e}(\theta)}=\boldsymbol{\upsilon}-\mathbf{y}$. Note the $\mathcal{S}(\boldsymbol{\upsilon})=\sigma(\boldsymbol{\upsilon})\left(\mathbf{1}-\sigma(\boldsymbol{\upsilon})\right)$, $e^{\boldsymbol{\upsilon}}$ and $\frac{\mathbf{y}}{e^{\boldsymbol{\upsilon}}}$ are *applied entrywise*.

Table 1: Some derived ODE systems of reparameterized ML model (1) using Theorem 1 w.r.t. different $L(\cdot)$. See corresponding sections for *rigorous* model definitions.

Lemma 1. *Suppose U is an invertible matrix with shape $m\times m$, and V is a $n\times n$ matrix. The inverse matrix of $\left[\begin{array}{cc}U & Z^T\\ Z & V\end{array}\right]$ equals to $\left[\begin{array}{cc}U^{-1}+U^{-1}Z^T\widetilde{H}^{-1}ZU^{-1} & -U^{-1}Z^T\widetilde{H}^{-1}\\ -\widetilde{H}^{-1}ZU^{-1} & \widetilde{H}^{-1}\end{array}\right]$, if $\widetilde{H}=V-ZU^{-1}Z^T$ is an invertible Schur complement.*

At each query of the ODE solver, we initialize a sweeping tableau as $\left[\begin{array}{c\|c} U & *\\ \hline Z & V\end{array}\right]$. Further sweeping of diagonal entries indexed by $\mathcal{A}$ yields $\left[\begin{array}{c\|c} -U^{-1} & *\\ \hline ZU^{-1} & \widehat{K}/U\end{array}\right]$, from which we can observe key elements appear in sweeping tableau. Combining with Lemma 1, the query complexity downgrade to $\mathcal{O}(|\mathcal{A}|\cdot|\mathcal{A}\cup\mathcal{E}|^2)+\mathcal{O}(|\mathcal{E}|^3)$. Compared to direct inversion, it also decreases a $\mathcal{O}(|\mathcal{A}\cup\mathcal{E}|^2)$ storage space.

5.3 Optimizing the Query Times

The length of solving interval $\left[\underline{\theta},\ \bar{\theta}\ \right]$ has a bearing on different varieties of φ_θ. A much shorter interval founds a significant decline of the total query times of an ODE solver. Here we give a simple example to illustrate it. Considering $\varphi_\theta^{(1)}=\theta^{\frac{1}{p}}$ and $\varphi_\theta^{(2)}=k\theta^{\frac{1}{p}}$ ($k\in\mathbb{Z}_+$). By calculating the $\underline{\theta}$ and $\bar{\theta}$, we know that the latter makes the length of solving interval become $\frac{1}{k^p}$ times of the former's. On the other hand, it should be emphasized that an over-complicated φ_θ will increase lots of difficulties in theoretical analysis, and the extremely short solution interval may tend to cause numerical instability. Note that the optimization of query times happens on the theoretical design stage of an ODE system, hence it's different from techniques in Sec. 5.1.

6 Examples of `DFLS`

Our examples are intended to illuminate the mechanics of `DFLS` and showcase its versatility. Due to the space limit, we provide greater details of implementation in our Appendix B, while the ODE systems of each subsection are displayed in Table 1. The following two theorems can help us analysis the ODE systems as mentioned in Sec. 4.

Theorem 2. *The optimal solution $\left\{\beta(\theta)|\theta\in\left[\underline{\theta},\ \bar{\theta}\ \right]\right\}$ in (5), (6), (7) or (8) is continuous w.r.t. θ.*

Theorem 3. *We have $\widehat{\mathcal{K}}$ in (5), (6), (7) or (8) is a real symmetric matrix.*

6.1 Group Lasso

Group Lasso (Yuan and Lin 2006) leads coefficient that only contains a few of wanted groups rather than sparsity in individual elements. We focus on group regularized optimization problem as

$$\min_{\beta} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\beta\|^2+\lambda\sum_{g\in G}\sqrt{d_g}\,\|\beta^g\|_2\,. \tag{5}$$

There has a total of G groups, d_g is the number of features in g-th group. We compute the (sub)gradient of regularization term as $\partial\sum_{g\in G}\|\beta^g\|_2 = \begin{cases}\frac{\beta^g}{\|\beta^g\|_2} & \text{if } \beta^g\neq\mathbf{0}\\ \in\{u:\|u\|_2\leqslant 1\} & \text{if } \beta^g=\mathbf{0}\end{cases}$. There follows $\nabla^2\mathcal{R}(\beta_{\mathcal{A}\cup\mathcal{E}}) = diag(B_i)$, where $B_g=\frac{\sqrt{d_g}}{\|\beta^g\|^3}\left(\|\beta^g\|^2\mathcal{I}-\beta^g\beta^{g\top}\right), g\in\mathcal{A}\cup\mathcal{E}$.

Segmentation First of all, we define the ξ_g as $\mathbf{X}_g^T(\mathbf{y}-\boldsymbol{\upsilon})$, where $g=1,\cdots,G$ denotes g-th predefined group. A non-zero β^g will decay to 0 as $\|\xi_g\|\leqslant\lambda\sqrt{d_g}, g\in\mathcal{A}$ (or $\varphi_\theta\|\xi_g\|\leqslant\lambda\sqrt{d_g}, g\in\mathcal{E}$) is found. Based on the continuity of solutions given by ODEs solver, we can set $\beta^g=0$ ($g\in\mathcal{A}\cup\mathcal{E}$) by the time it reaches the opposite sign and remove index g from $\mathcal{A}$. Alternatively, when the norm value of $\xi_g, g\in\overline{\mathcal{A}}$ (or $\varphi_\theta\|\xi_g\|, g\in\overline{\mathcal{E}}$) comes to $\lambda\sqrt{d_g}$, the g-th group becomes active and will be added into the corresponding active set. With that, we can keep observation on the KKT conditions for each g in $\overline{\mathcal{A}}\cup\overline{\mathcal{E}}$ while solving the ODE system to estimate whether a new segment is encountered.

Extension 1 (Overlapped Group Lasso (Jacob, Obozinski, and Vert 2009))**.** *Please refer to Appendix C.3.*

Extension 2 (ℓ_p-norm Penalization (Lu 2014))**.** *We consider the regularized $L(\cdot)$ where $R=\|\beta\|_p^p$, in which $0<p<1$ gives non-convex regularization and $p>1$ gives a convex regularizer. Our* DFLS *can also be extended to them.*

6.2 Ridge Logistic Estimator

We consider the ridge logistic estimator (Le Cessie and Van Houwelingen 1992) for classification where label $y_i\in\{0,1\}$. As an optimization problem, binary class ℓ_2-penalized logistic regression minimizes the following cost function

$$\min_{\beta}\ \frac{1}{2}\beta^T\beta+C\sum_{i=1}^{n}-y_i\log\left(h_\beta(\mathbf{X}_i)\right)-(1-y_i)\log\left(1-h_\beta(\mathbf{X}_i)\right), \tag{6}$$

where $h_\beta(\mathbf{X}_i)=\frac{1}{1+e^{-\mathbf{X}_i\beta}}$ is hypothesis function. By solving the ODE system in Table 1, the feature-wise online learning for logistic estimator can be realized here.

6.3 Poisson Regression

The Poisson regression (Frome 1983) is the problem (7) where one minimises Poisson deviance. Our theory and practice indicate that the DFLS also fits a series of generalized linear models like (7) and (8).

$$\min_{\beta}\frac{1}{n}\sum_{i=1}^{n}\left[y_i\log\frac{y_i}{\exp\left(\mathbf{X}_i\beta\right)}-\left(y_i-\exp\left(\mathbf{X}_i\beta\right)\right)\right]+\lambda\|\beta\|^2\,. \tag{7}$$

Dataset	Size	Dimension
Yolanda	400,000	101
BNG(libras_move)	1,000,000	90
satellite_image	6435	37
BNG(wisconsin)	1,000,000	33
cpu_act	8192	22

Table 2: Summary of the real-world datasets in experiments.

Extension 3 (Gamma Regression (Prentice 1974))**.** *Similarly, we minimizes the gamma deviance as*

$$\min_{\beta}\frac{1}{n}\sum_{i=1}^{n}\left[\log\frac{\exp\left(\mathbf{X}_i\beta\right)}{y_i}+\frac{y_i}{\exp\left(\mathbf{X}_i\beta\right)}-1\right]+\lambda\|\beta\|^2\,. \tag{8}$$

6.4 Extension on Non-linear Pattern

Extension 4 (Polynomial Regression (Max and Burkhart 1976))**.** *Polynomial regression is capable of fitting nonlinear polynomial functions of the input data, and it's operated by generating a new feature space consisting of all polynomial combinations of the original features. Suppose $\vec{\mathbf{X}}$ is generated polynomial features and is placed in (2), the online learning pipeline of* DFLS *is the same as it used to be.*

Extension 5 (Spline Interpolation (Schoenberg 1973))**.** *Please refer to Appendix D.2.*

Extension 6 (Random Fourier Features (Rahimi and Recht 2007))**.** *Please refer to Appendix D.3.*

7 Numerical Experiments

In this section, we provide experimental results on standard datasets to validate our theoretical results.

Baselines & Datasets We perform a comparison against the following batch training methods. 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[3] (O'donoghue and Candes 2015). The batch training algorithm for other algorithms uses stochastic gradient descent (SGD) (Bottou 2010) to optimize the model. *Other implementation details are in Appendix E*. Table 2 summarizes the datasets we used, which are available on OpenML (Vanschoren et al. 2014).

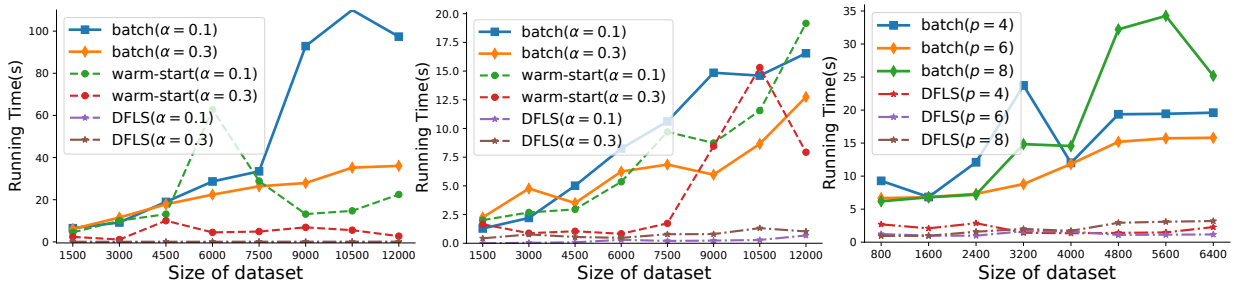
Accuracy Comparison To prove the practicability of DFLS, we compare the generalization performance of DFLS with batch re-training algorithms. We randomly select a subset of features in whole feature space to perform online learning simulations. In each round, we randomly select about 10% features (round down). Once selected, we delete the selected features from current space and add them back to the model in the next round[4]. After features are selected and fixed, all the experiments were conducted over 10 times for *each round*. Numerical results are shown in Table 3, from which we can observe that the generalization errors of DFLS

[3]https://github.com/yngvem/group-lasso

[4]Note the "$(+)$" and "$(-)$ " round in Table 3 appear alternately.

Method	rounds										Dataset
	$1_{(-)}$	$2_{(+)}$	$3_{(-)}$	$4_{(+)}$	$5_{(-)}$	$6_{(+)}$	$7_{(-)}$	$8_{(+)}$	$9_{(-)}$	$10_{(+)}$	
Group Lasso†	11.71	10.07	11.55	10.07	10.95	10.07	12.66	10.07	12.23	10.07	Yolanda
Group Lasso‡	11.71	10.07	11.55	10.07	10.95	10.07	12.66	10.07	12.23	10.07	
E_{max}	1e-8	4e-7	5e-7	3e-8	4e-7	3e-7	4e-7	3e-7	6e-7	6e-7	
ℓ_p OLS $(p=0.5)$†	5.90	5.06	5.86	5.06	5.67	5.06	4.69	5.06	6.72	5.06	BNG(libras_move)
ℓ_p OLS $(p=0.5)$‡	5.90	5.06	5.86	5.06	5.67	5.06	4.69	5.06	6.72	5.06	
E_{max}	8e-7	1e-6	5e-7	1e-7	4e-7	4e-7	4e-7	2e-7	1e-7	2e-7	
ℓ_p OLS $(p=4)$†	4.39	4.74	5.10	4.74	5.86	4.74	4.84	4.74	4.45	4.74	BNG(libras_move)
ℓ_p OLS $(p=4)$‡	6.39	4.74	5.10	4.74	5.86	4.74	4.84	4.74	4.45	4.74	
E_{max}	4e-8	9e-7	2e-7	9e-7	3e-7	4e-7	7e-7	2e-7	8e-6	1e-6	
Logistic Estimator†	1.64	1.96	1.05	1.96	1.38	1.96	0.91	1.96	2.01	1.96	satellite_image
Logistic Estimator‡	1.64	1.96	1.05	1.96	1.38	1.96	0.91	1.96	2.00	1.96	
E_{max}	5e-7	5e-7	9e-7	4e-8	6e-7	4e-7	2e-7	9e-6	2e-7	2e-7	
Poisson Regression†	32.01	30.15	31.78	30.15	31.02	30.15	30.25	30.15	28.99	30.15	BNG(wisconsin)
Poisson Regression‡	32.01	30.15	31.78	30.15	31.02	30.15	30.25	30.15	28.99	30.15	
E_{max}	9e-7	4e-8	4e-8	5e-8	4e-8	8e-8	2e-7	5e-7	6e-8	2e-7	
Gamma Regression†	11.95	10.11	12.85	10.11	10.00	10.11	10.64	10.11	10.10	10.11	cpu_act
Gamma Regression‡	11.95	10.11	12.85	10.11	10.00	10.11	10.64	10.11	10.10	10.11	
E_{max}	3e-8	9e-8	4e-7	8e-8	2e-7	6e-7	7e-7	9e-7	8e-7	2e-7	

Table 3: Numerical results of root mean square error. No variances are recorded since they are too small to be kept. The OLS = "Ordinary Least Squares" and $E_{max} = \max \|\hat{\beta} - \beta^*\|$ records maximum error in each trail, where β^* is the optimal solution returned by batch training and $\hat{\beta}$ is given by `DFLS`. † indicates the online `DFLS` method. ‡ indicates using the re-training approaches. At each round, $(+)$ denotes feature adding and $(-)$ denotes removing features from current feature space.



(a) group Lasso (b) Poisson regression (c) ℓ_p OLS $(\lambda = 0.05)$

Figure 3: Efficiency comparison of our `DFLS` and existing algorithms. The α is regularization parameter (equals to λ) and p is hyperparameter defined in Extension 2. When using warm start, the batch training will not re-initialize parameters.

consist with the results of batch re-training baselines. The simulations are also conducted on the *non-linear* patterns of Section 6.4 in Appendix D. These comparisons prove that our algorithms enjoy a strong accuracy guarantee.

Efficiency Comparison We evaluate average processing times in ten runs when executing one round of online learning (feature numbers $d' = 5$) using our `DFLS` and other methods under diverse data scales. The results of the first five trails are graphically described in Figure 3, while we list the more comparison results in Appendix F.2. Compared with retraining a new learning model from scratch, `DFLS` can exploit both the knowledge of the existing model and the newly arrived features with less training time. We also plot the learning curves in Appendix F.5 for baseline fairness. The Appendix F.4 shows **a contrary case** that our algorithm is inefficient in exceptional high-dimensional cases due to the increased time on solving the right hand of (3).

8 Conclusion

At this work, we propose an efficient online learning algorithm with accuracy guarantee and present practical guides to solving some important learning problems. Simulation studies show that the proposed `DFLS` can substantially ease the computational cost while keeping the model's accuracy.

Acknowledgments

Bin Gu was partially supported by the National Natural Science Foundation of China (No:61573191).

References

Agarwal, S.; Saradhi, V. V.; and Karnick, H. 2008. Kernel-based online machine learning and support vector reduction. *Neurocomputing*, 71(7-9): 1230–1237.

Alagurajah, J.; Yuan, X.; and Wu, X. 2020. Scale invariant learning from trapezoidal data streams. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 505–508.

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.

Beyazit, E. 2021. *Online Learning with Varying Feature Spaces*. Ph.D. thesis, University of Louisiana at Lafayette.

Beyazit, E.; Alagurajah, J.; and Wu, X. 2019. Online learning from data streams with varying feature spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3232–3239.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, 177–186. Springer.

Bottou, L.; and Cun, Y. 2003. Large scale online learning. *Advances in neural information processing systems*, 16.

Dekel, O.; Shamir, O.; and Xiao, L. 2010. Learning to classify with missing and corrupted features. *Machine learning*, 81(2): 149–178.

Ehrlich, L. W. 1967. A modified Newton method for polynomials. *Communications of the ACM*, 10(2): 107–108.

Fontenla-Romero, Ó.; Guijarro-Berdiñas, B.; Martinez-Rego, D.; Pérez-Sánchez, B.; and Peteiro-Barral, D. 2013. Online machine learning. In *Efficiency and Scalability Methods for Computational Intellect*, 27–54. IGI Global.

Freund, R. M.; Grigas, P.; and Mazumder, R. 2013. Incremental Forward Stagewise Regression: Computational Complexity and Connections to LASSO. *URL http://www. esat. keluwen. be/sista/ROKS2013. Available on-line*.

Frome, E. L. 1983. The analysis of rates using Poisson regression models. *Biometrics*, 665–674.

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

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.

Goodnight, J. H. 1979. A tutorial on the SWEEP operator. *The American Statistician*, 33(3): 149–158.

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.

He, Y.; Wu, B.; Wu, D.; Beyazit, E.; Chen, S.; and Wu, X. 2019. Online learning from capricious data streams: a generative approach. In *International Joint Conference on Artificial Intelligence Main track*.

He, Y.; Wu, B.; Wu, D.; Beyazit, E.; Chen, S.; and Wu, X. 2020. Toward mining capricious data streams: A generative approach. *IEEE transactions on neural networks and learning systems*, 32(3): 1228–1240.

He, Y.; Yuan, X.; Chen, S.; and Wu, X. 2021. Online Learning in Variable Feature Spaces under Incomplete Supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4106–4114.

Hoerl, A. E.; and Kennard, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1): 55–67.

Hofleitner, A.; Rabbani, T.; El Ghaoui, L.; and Bayen, A. 2013. Online Homotopy Algorithm for a Generalization of the LASSO. *IEEE Transactions on Automatic Control*, 58(12): 3175–3179.

Hofleitner, A.; Rabbani, T.; Rafiee, M.; El Ghaoui, L.; and Bayen, A. 2014. Learning and estimation applications of an online homotopy algorithm for a generalization of the LASSO. *Discrete & Continuous Dynamical Systems-S*, 7(3): 503.

Hou, B.-J.; Yan, Y.-H.; Zhao, P.; and Zhou, Z.-H. 2021. Storage Fit Learning with Feature Evolvable Streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7729–7736.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2017. Learning with feature evolvable streams. *Advances in Neural Information Processing Systems*, 30.

Hou, C.; and Zhou, Z.-H. 2017. One-pass learning with incremental and decremental features. *IEEE transactions on pattern analysis and machine intelligence*, 40(11): 2776–2792.

Hu, C.; Chen, Y.; Peng, X.; Yu, H.; Gao, C.; and Hu, L. 2018a. A novel feature incremental learning method for sensor-based activity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 31(6): 1038–1050.

Hu, X.; Zhou, P.; Li, P.; Wang, J.; and Wu, X. 2018b. A survey on online feature selection with streaming features. *Frontiers of Computer Science*, 12(3): 479–493.

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.

Lange, K.; Chambers, J.; and Eddy, W. 1999. *Numerical analysis for statisticians*, volume 2. Springer.

Le Cessie, S.; and Van Houwelingen, J. C. 1992. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(1): 191–201.

Li, D.; and Gu, B. 2022. Chunk Dynamic Updating for Group Lasso with ODEs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7): 7408–7416.

Li, H.; Wu, X.; Li, Z.; and Ding, W. 2013. Group feature selection with streaming features. In *2013 IEEE 13th International Conference on Data Mining*, 1109–1114. IEEE.

Little, R. J.; and Rubin, D. B. 2019. *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

Lu, Z. 2014. Iterative reweighted minimization methods for l_p regularized unconstrained nonlinear programming. *Mathematical Programming*, 147(1): 277–307.

Max, T. A.; and Burkhart, H. E. 1976. Segmented polynomial regression applied to taper equations. *Forest Science*, 22(3): 283–289.

Nocedal, J.; and Wright, S. J. 1999. *Numerical optimization*. Springer.

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

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113: 54–71.

Prentice, R. L. 1974. A log gamma model and its maximum likelihood estimation. *Biometrika*, 61(3): 539–544.

Rahimi, A.; and Recht, B. 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20.

Schaefer, R.; Roi, L.; and Wolfe, R. 1984. A ridge logistic estimator. *Communications in Statistics-Theory and Methods*, 13(1): 99–113.

Schoenberg, I. J. 1973. *Cardinal spline interpolation*. SIAM.

Vanschoren, J.; Van Rijn, J. N.; Bischl, B.; and Torgo, L. 2014. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2): 49–60.

Wang, J.; Wang, M.; Li, P.; Liu, L.; Zhao, Z.; Hu, X.; and Wu, X. 2015. Online feature selection with group structure analysis. *IEEE Transactions on Knowledge and Data Engineering*, 27(11): 3029–3041.

Wang, J.; Zhao, P.; Hoi, S. C.; and Jin, R. 2013a. Online feature selection and its applications. *IEEE Transactions on knowledge and data engineering*, 26(3): 698–710.

Wang, J.; Zhao, Z.-Q.; Hu, X.; Cheung, Y.-M.; Wang, M.; and Wu, X. 2013b. Online group feature selection. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 1757–1763.

Watson, L. T. 1986. Numerical linear algebra aspects of globally convergent homotopy methods. *SIAM review*, 28(4): 529–545.

Watson, L. T. 2001. Theory of globally convergent probability-one homotopies for nonlinear programming. *SIAM Journal on Optimization*, 11(3): 761–780.

Wu, X.; Yu, K.; Ding, W.; Wang, H.; and Zhu, X. 2012. Online feature selection with streaming features. *IEEE transactions on pattern analysis and machine intelligence*, 35(5): 1178–1192.

Wu, X.; Yu, K.; Wang, H.; and Ding, W. 2010. Online streaming feature selection. In *ICML*.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; and Fu, Y. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 374–382.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2018. Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations*.

You, D.; Wu, X.; Shen, L.; Chen, Z.; Ma, C.; and Deng, S. 2018. Online feature selection for streaming features with high redundancy using sliding-window sampling. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, 205–212. IEEE.

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, F. 2006. *The Schur complement and its applications*, volume 4. Springer Science & Business Media.

Zhang, Q.; Zhang, P.; Long, G.; Ding, W.; Zhang, C.; and Wu, X. 2015. Towards mining trapezoidal data streams. In *2015 IEEE International Conference on Data Mining*, 1111–1116. IEEE.

Zhang, Q.; Zhang, P.; Long, G.; Ding, W.; Zhang, C.; and Wu, X. 2016. Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(10): 2709–2723.

Zhang, Z.-Y.; Zhao, P.; Jiang, Y.; and Zhou, Z.-H. 2020. Learning with feature and distribution evolvable streams. In *International Conference on Machine Learning*, 11317–11327. PMLR.

Zhou, P.; Li, P.; Zhao, S.; and Wu, X. 2020. Feature interaction for streaming feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10): 4691–4702.