OPT2022: 14th Annual Workshop on Optimization for Machine Learning

An Accuracy Guaranteed Online Solver for Learning in Dynamic Feature Space

Diyang Li DIYOUNG.LEE@GMAIL.COM
Nanjing University of Information Science & Technology

Bin Gu BIN.GU@MBZUAI.AC.AE
MBZUAI

Abstract

We study the problem of adding or deleting features of data from machine learning models trained using empirical risk minimization. Our focus is on algorithms in an online manner which is capable for a more general regularization term, and present practical guides to two classical regularizers, *i.e.*, the group Lasso and ℓ_p-norm regularizer. Across a variety of benchmark datasets, our algorithm improves upon the runtime of prior methods while maintaining the *same* generalization accuracy.

1. Introduction

In traditional machine learning, we normally assume that the set of features is known and fixed. However, it's unrealistic to many real-world scenarios [5, 6, 17]. To address this, incremental (or decremental) algorithm [8] was proposed to incorporate or eliminate features, which is a promising family of learning algorithms due to the updated model can refine its knowledge without re-training from scratch [16]. In supervised learning, an extensive number of algorithms essentially solve the regularized minimization problem, and for a linear model, it reads

$$\min_{\boldsymbol{w}} \sum_i \mathcal{L}\left(x_i, y_i; \boldsymbol{w}\right) + \alpha \mathcal{R}(\boldsymbol{w}), \tag{1}$$

where $\mathcal{R}(\cdot)$ is the regularization term and $\mathcal{L}\left(\cdot\right)$ denotes the loss function. The $\boldsymbol{w}$ represents learnable parameters inside the model. Although there have been many efforts made to incorporate or eliminate features in an exact way, they are still limited to primitive regularization terms such as ℓ_1 and ℓ_2 norms in (1). Specifically, the Lasso regularizer has been solved in [7] using forward stagewise regression. Other simple regularization like ℓ_2-norm leads to analytical solutions [9], hence result can be computed directly in the new feature space. To fill this blank, we propose a learning system that handles a class of regularized linear statistical models with dynamic features, termed `Dynamic Feature Learning System`(`DFLS`), where the updated model *is exactly the same as a model trained from scratch* using the entire new feature space. Extensive experimental results validate its efficiency compared to the existing training algorithm.

2. Preliminaries

Given the dataset $\mathcal{D} = \{X \in \mathbb{R}^{n\times m}, \boldsymbol{y} \in \mathbb{R}^n\}$ with n observations and each observation has m features. The $\boldsymbol{y}_i \in \mathbb{R}$ for the regression problem[1] and $\boldsymbol{y}_i \in \{-1, 1\}$ for the binary classification.

1. The bias term in target can be introduced by expanding X with one column whose all elements are 1.

Assumption 1 *We assume that the loss function $\mathcal{L}(\cdot)$ and regularization item $\mathcal{R}(\cdot)$ in (1) are piece-wise (or element-wise) second-order differentiable.*

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

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

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

Objective Rewriting To simplify notations we organize (or partition) the traning data as $\left[X, \widetilde{X}\right]$ and coefficient vector $\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix}$ by two types of features, where $\widetilde{X} \in \mathbb{R}^{n\times m'}$ includes m' features we intend to add into (or delete from) the original feature set. The $\mathcal{A}$ and $\mathcal{E}$ are active sets that containing indices of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ that $\boldsymbol{w}_{ij} \neq 0$ $(i = 1, 2)$, so as to store active components, *i.e.*, we have $\boldsymbol{w}_1 = \left(\boldsymbol{w}_{\mathcal{A}}^T, \boldsymbol{w}_{\overline{\mathcal{A}}}^T\right)^T$, $\boldsymbol{w}_2 = \left(\boldsymbol{w}_{\mathcal{E}}^T, \boldsymbol{w}_{\overline{\mathcal{E}}}^T\right)^T$, where elements in $\boldsymbol{w}_{\overline{\mathcal{A}}}, \boldsymbol{w}_{\overline{\mathcal{E}}}$ remains 0. We denote a point where the active set changes a "*transition point*". Now we introduce the following concrete formulation[3]

$$\min_{\boldsymbol{w}\in\mathbb{R}^{m+m'}} \frac{1}{2n} \left\| \left[X, \phi_\theta \widetilde{X}\right] \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix} - \boldsymbol{y} \right\|_F^2 + \alpha \mathcal{R}\left(\begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix} \right), \tag{2}$$

where $\alpha > 0$ is regularization parameter to balance the prediction loss and penalty. A rational ϕ_θ in (2) can be used to reparameterize the addition or subtraction procedure of a chunk feature $\widetilde{X}$, which satisfies ***(i).***ϕ_θ are monotonically increasing function w.r.t. θ in closed interval $[0, 1]$. ***(ii).***ϕ_θ have the properties of continuity and smoothness, meanwhile $\phi_\theta|_{\theta=0} = 0$, $\phi_\theta|_{\theta=1} = 1$ should be kept. Here we can follow the solution path of $\boldsymbol{w}$ when varying θ from 0 to 1 to add $\widetilde{X}$. Conversely, as we vary θ from 1 to 0, the elimination of features in $\widetilde{X}$ can then be performed in a similar way.

Investigate Optimality Conditions The KKT point [3] of the problem (2) can be obtained by

$$\frac{1}{n} \begin{bmatrix} X^T \\ \phi_\theta \widetilde{X}^T \end{bmatrix} \underbrace{\left[X\boldsymbol{w}_1 + \phi_\theta \widetilde{X}\boldsymbol{w}_2 - \boldsymbol{y}\right]}_{\text{denotes as } \widehat{\boldsymbol{e}(\theta)}} + \alpha \begin{bmatrix} \boldsymbol{u}_{\boldsymbol{w}_1} \\ \boldsymbol{u}_{\boldsymbol{w}_2} \end{bmatrix} = \mathbf{0}, \tag{3}$$

where $\boldsymbol{u} \in \partial\mathcal{R}(\boldsymbol{w})$ is subgradient representation.

2. To simplify notations we assume the new $\widetilde{X}$ appears last.
3. We mainly study a frequently-used square loss function, and others can be derived following our analysis.

3. Accuracy Guaranteed Online Solver

3.1. Algorithm Outline

We first pointed out that there potentially exists two stages for tracking the optimal $\boldsymbol{w}$. The proposed updating stages are summarized as ***(i).***The solution path of θ would remain constant untill the first transition point $\widetilde{\theta}$ is met. We compute its value in closed form and update $\boldsymbol{w}$ directly. ***(ii).***It's non-trivial for analytical solutions of $\boldsymbol{w}(\theta)$ in (3), especially some complex penalizations. We use the differential equation to gain the desired solution path[4] untill $\theta = 1$. The usage of each stage depends on the properties of regularization term. The criterion of whether to use each phase are summarized in Table 1. The decremental learning can be viewed as an inverse process for tracking the path from $\theta = 1$ to $\theta = 0$.

Property of $\mathcal{R}(\cdot)$	Phase I Update	Phase II Update
Sparsity	✔	depends on $\widehat{\theta}$
Non-Sparsity	✗	✔

Table 1: The choice of two stages in updating process. A tick signifies necessity and vice versa.

3.2. Phase I Update

Due to the existence of $\mathcal{R}(\boldsymbol{w})$, at first (*i.e.* $\theta = 0$) all coefficients in $\boldsymbol{w}_2$ will converges to 0. As $\boldsymbol{w}_2|_{\theta=0} = \mathbf{0}$, by plugging it into (3) we have (4) and (5). Considering inactive components in (5), by the properties of subgradient we get (6). Since ϕ_θ is continuous w.r.t. θ, we can conclude that there exists a $\widetilde{\theta} > 0$ such that the inequality (6) holds in $[0, \widetilde{\theta})$. The solution path would be smooth before approaches a transition point, and non-smooth graphically at transition points. From (4) we can see the optimal $\boldsymbol{w}_1$ has nothing to do with θ when $\theta \in [0, \widetilde{\theta})$, it will be a constant vector till the θ reaches $\widetilde{\theta}$. Assume there are one component becomes active, we calculate the first encountered transition point $\widetilde{\theta}$ in exact solution

$$\frac{1}{n} X^T (X\boldsymbol{w}_1 - \boldsymbol{y}) + \alpha \boldsymbol{u}_{\boldsymbol{w}_1} = \mathbf{0} \tag{4}$$

$$\frac{\phi_\theta}{n} \cdot \widetilde{X}^T (X\boldsymbol{w}_1 - \boldsymbol{y}) + \alpha \boldsymbol{u}_{\boldsymbol{w}_2} = \mathbf{0} \tag{5}$$

$$\alpha \|\boldsymbol{u}_{\boldsymbol{w}_{\overline{\mathcal{E}}}}\| \leq \frac{\phi_\theta}{n} \cdot \|\widetilde{X}_{\overline{\mathcal{E}}}^T (X\boldsymbol{w}_1 - \boldsymbol{y})\| \tag{6}$$

$$\phi_\theta|_{\theta=\widetilde{\theta}} = \min \left\{ \alpha n \cdot \frac{\|\nabla \mathcal{R}(\boldsymbol{w}_i)\|}{\left\| \widetilde{X}_i^T (y - X_{\mathcal{A}} \boldsymbol{w}_{\mathcal{A}}) \right\|} \right\}, \text{ for } i \in \overline{\mathcal{E}}. \tag{7}$$

Then we update

$$\boldsymbol{w}|_{\theta=\widehat{\theta}} \leftarrow \boldsymbol{w}|_{\theta=0}, \quad \text{where } \widehat{\theta} = \min \left\{ \widetilde{\theta}, 1 \right\}.$$

If $\widehat{\theta} = 1$, then we can skip the second stage and finish the algorithm. From expression in (7), we know that the location of the first transition point is proportional to parameter α, which makes our approach very efficient when α in relatively large quantities.

Remark 1 *When we use decremental learning, there are no analytical solution for $\widehat{\theta}$. The boundary condition for Phase I (Phase II) is that whether $\mathcal{E} = \varnothing$ (or $\boldsymbol{w}_2 = \mathbf{0}$).*

4. The Picard–Lindelöf theorem [12] straightforwardly proves that the solution (path) of (2) is unique and can be extended to the nearest boundary of θ (*i.e.*, until the violater(s) of active sets $\mathcal{A}$ or $\mathcal{E}$ appears).

3.3. Phase II Update

Theorem 2 *When the active sets $\mathcal{A}$ and $\mathcal{E}$ are fixed (in a subinterval of $[0,1]$), the solution path of $\boldsymbol{w}$ satisfies the following first-order ODE system*

$$\widehat{\mathcal{K}} \begin{bmatrix} \frac{d\boldsymbol{w}_{\mathcal{A}}}{d\theta} \\ \frac{d\boldsymbol{w}_{\mathcal{E}}}{d\theta} \end{bmatrix} = -\frac{1}{n} \begin{bmatrix} \phi'_\theta \cdot X_{\mathcal{A}}^T \widetilde{X}_{\mathcal{E}} \boldsymbol{w}_{\mathcal{E}} \\ \widetilde{X}_{\mathcal{E}}^T \left(\phi'_\theta \widehat{\boldsymbol{e}(\theta)} + \phi_\theta \phi'_\theta \widetilde{X}_{\mathcal{E}} \boldsymbol{w}_{\mathcal{E}} \right) \end{bmatrix}, \tag{8}$$

where $(\cdot)'$ is the derivative note. The key matrix $\widehat{\mathcal{K}}$ has the form of $\begin{bmatrix} U & Z^T \\ Z & V \end{bmatrix}$, where $U = \frac{1}{n} X_{\mathcal{A}}^T X_{\mathcal{A}} + \alpha \nabla^2 \mathcal{R}(\boldsymbol{w}_{\mathcal{A}})$, $V = \frac{\phi_\theta^2}{n} \cdot \widetilde{X}_{\mathcal{E}}^T \widetilde{X}_{\mathcal{E}} + \alpha \nabla^2 \mathcal{R}(\boldsymbol{w}_{\mathcal{E}})$ and $Z = \frac{\phi_\theta}{n} \cdot \widetilde{X}_{\mathcal{E}}^T X_{\mathcal{A}}$.

Theorem 3 *We have $\widehat{\mathcal{K}}$ is a real symmetric matrix.*

The proof of Theorems are listed in Appendix A. With Theorem 3, we can use the low-rank update or Woodbury formulae [14], which essentially solves (8) with at most $\mathcal{O}(|\mathcal{A} \cup \mathcal{E}|^2)$ work. As we obtaining the $\boldsymbol{w}|_{\theta=0}$ and $\frac{d\boldsymbol{w}_{\mathcal{A}\cup\mathcal{E}}}{d\theta}$, by solving the initial-value problem numerically with ODE solvers, the solution path regarding to θ can be computed swiftly before active sets $\mathcal{A}$ or $\mathcal{E}$ changes. Some ***visualization examples*** of solution path are listed in Appendix C.

Handling Transition Point The emergence of a transition point is owing to indices change in $\mathcal{A}$ or $\mathcal{E}$, which is characterized by the variation of optimality conditions (3). Meanwhile the ODE wouldn't work due to they are non-differentiable points. We can keep observation on the optimality conditions while solving the path to estimate if there exists transition point(s), and *update the active sets $\mathcal{A}$ and $\mathcal{E}$ accordingly*. The violator(s) could be easily identified by the thresholding conditions. For non-sparsity penalizations, there will be no changes for $\mathcal{A}$ or $\mathcal{E}$, hence $\frac{d\boldsymbol{w}}{d\theta}$ is continuous of θ on $(0,1)$.

4. `DFLS` Implementation

In this part, we select the transformation as $\phi_\theta \overset{\text{def}}{=} \theta$. Due to the space limit, we defer the algorithmic steps and feature *decremental* learning procedure in our Appendix B, Appendix E, respectively.

Group Lasso Group Lasso [18] has been successful in many practical applications [4, 10]. There has a total of G groups, d_g is the number of features in g-th group. We focus on group regularized optimization problem as (9).

$$\min_{\boldsymbol{w}} \frac{1}{2n} \|X\boldsymbol{w} - \boldsymbol{y}\|^2 + \alpha \sum_{g=1}^{G} \sqrt{d_g} \|\boldsymbol{w}^g\|_2 \tag{9}$$

ℓ_p Penalization Optimization method towards ℓ_p-norm penalty has been studied extensively by researchers [11]. Here we present the case of $p > 1$, which gives a convex setting of $\mathcal{R}(\cdot)$. Our learning objective is shown as (10).

$$\min_{\boldsymbol{w}} \frac{1}{2n} \|X\boldsymbol{w} - \boldsymbol{y}\|^2 + \alpha \|\boldsymbol{w}\|_p^p \tag{10}$$

5. Simulation Experiments

5.1. Experimental Setup

In this part, we provide experiments on real-world datasets to validate our theoretical results. Our experiments are delivered from the following two perspectives. Table 2 summarizes these datasets we used.

Dataset	Size	Dimension
Yolanda	50000	100
BNG(libras_move)	20000	90
satellite_image	6435	37
BNG(wisconsin)	50000	32
cpu_act	8192	22
MR#1	100	500
MR#2	100	1000

Table 2: Summary of the real-world dasetsets and synthetic data (MR#1∼3) in experiments.

Accuracy To prove the practicability of proposed approach, we compare the generalization performance of `DFLS` with batch algorithms. We conduct training tasks on real-world data and randomly select about 20% features in whole feature space to perform dynamic updating. In detail, we utilize the well-trained model to add or remove around 20% features and compare the average Root Mean Squared Error (aRMSE) between several methods in ten runs. The batch algorithms re-train the model with both cold-start and warm-start (i.e., the subsequent calls will not re-initialise parameters). Numerical results are shown in Table 3 and Table 4, while we put the results of *decremental* learning in Appendix E.

Efficiency We evaluate average processing times in five runs when executing one dynamic updating (feature numbers $m' = 5$) using our `DFLS` and other methods under diverse data scales. Additionally, we test the training time on $m \gg n$ cases based on MR#1∼2. 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.

Baselines Setup We adopt a widely-used batch training algorithm of group Lasso using FISTA optimiser [1] with a gradient-based adaptive restarting scheme[5] [13]. The batch training algorithm for ℓ_p penalization uses Stochastic Gradient Descent (SGD) [2] to optimize the model. *Other implementation details are in Appendix D.*

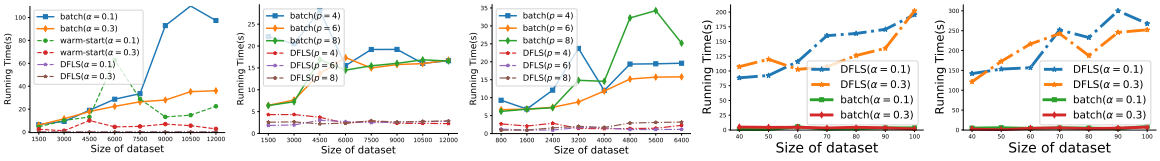
Dataset		Yolanda		BNG(libras_move)		satellite_image		BNG(wisconsin)		cpu_act	
G	α	`DFLS`	batch	`DFLS`	batch	`DFLS`	batch	`DFLS`	batch	`DFLS`	batch
I	0.2	10.02	10.02	4.03	4.03	1.55	1.55	29.91	29.91	10.11	10.11
	0.5	10.77	10.77	4.26	4.26	1.67	1.67	30.16	30.16	10.10	10.10
II	0.2	10.09	10.07	4.02	4.02	1.52	1.52	29.97	29.97	10.31	10.31
	0.5	10.50	10.50	4.24	4.24	1.63	1.63	29.96	29.96	10.46	10.46
III	0.2	10.61	10.61	4.38	4.38	1.88	1.88	30.69	30.69	9.98	9.98
	0.5	10.87	10.87	4.63	4.63	2.20	2.21	30.86	30.86	9.87	9.87

Table 3: Results of aRMSE when training the group Lasso with incremental features. Any variance less than 10^{-5} are omitted. `DFLS` and *batch* represent each iteration is trained with a chunk of new features, or using batch algorithm to retrain from scratch, respectively.

5. Code available at https://github.com/yngvem/group-lasso

Dataset		Yolanda		BNG(libras_move)		satellite_image		BNG(wisconsin)		cpu_act	
ℓ_p	α	DFLS	batch	DFLS	batch	DFLS	batch	DFLS	batch	DFLS	batch
ℓ_4	0.2	9.82	9.82	3.73	3.73	1.20	1.20	31.20	31.20	14.22	14.22
	0.5	9.91	9.91	3.74	3.74	1.22	1.22	31.66	31.66	14.88	14.86
ℓ_6	0.2	9.66	9.66	3.73	3.73	1.20	1.20	31.27	31.27	14.28	14.28
	0.5	9.90	9.89	3.74	3.74	1.20	1.21	31.91	31.90	14.94	14.94

Table 4: The aRMSE results of ℓ_p-norm regularized regression model with incremental features. Any variance less than 10^{-5} are omitted.



(a). group Lasso [Yolanda] (b). ℓ_p penalty [BNG(libras_move)] (c). ℓ_p penalty [satellite_image] (d). ℓ_p penalty [MR#1] (e). ℓ_p penalty [MR#2]

Figure 1: Efficiency comparison of different algorithms. The $\alpha = 0.2$ for ℓ_p penalty in (b) and (c). $[\cdot]$ indicates the name of dataset.

5.2. Results & Discussions

From above Table 3 and Table 4, our algorithm enjoys high precision, which is characterized by very closing to the aRMSE of batch training method. Moreover, Figure 1 clearly demonstrate that there exists a large time gap compared to other learning strategies while our DFLS keeping the very similar precision. This is due to the fact that our algorithm can track the path of θ without training the whole model in new feature space from scratch. Furthermore, the training time in Figure 1 (d). and (e). suggest our algorithm is inefficient compared to batch training in exceptional high-dimensional cases due to the increased time on solving the linear system in (8). Particularly, in $m \gg n$, the advantages of DFLS would gradually erased as m grows.

6. Conclusion

At this work we propose an efficient online solver via ODEs to perform features expansion and shrinkage for linear models, and present practical guides to solving group Lasso and ℓ_p-norm regularized regression. Simulation studies show that the proposed algorithm can substantially ease the computational cost while keeping the accuracy performance.

References

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

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Soumyadeep Chatterjee, Karsten Steinhaeuser, Arindam Banerjee, Snigdhansu Chatterjee, and Auroop Ganguly. Sparse group lasso: Consistency and climate applications. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 47–58. SIAM, 2012.

[5] Ofer Dekel, Ohad Shamir, and Lin Xiao. Learning to classify with missing and corrupted features. *Machine learning*, 81(2):149–178, 2010.

[6] Óscar Fontenla-Romero, Bertha Guijarro-Berdiñas, David Martinez-Rego, Beatriz Pérez-Sánchez, and Diego Peteiro-Barral. Online machine learning. In *Efficiency and Scalability Methods for Computational Intellect*, pages 27–54. IGI Global, 2013.

[7] Robert M Freund, P Grigas, and R Mazumder. Incremental forward stagewise regression: Computational complexity and connections to lasso. *URL http://www. esat. keluwen. be/sista/ROKS2013. Available on-line*, 2013.

[8] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (ESANN)*, 2016.

[9] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[10] Michael Lim and Trevor Hastie. Learning interactions via hierarchical group-lasso regularization. *Journal of Computational and Graphical Statistics*, 24(3):627–654, 2015.

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

[12] George Moseley Murphy. *Ordinary differential equations and their solutions*. Courier Corporation, 2011.

[13] Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

[14] Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1): 124–127, 1950.

[15] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL http://doi.acm.org/10.1145/2641190.2641198.

[16] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

[17] Dianlong You, Xindong Wu, Limin Shen, Zhen Chen, Chuan Ma, and Song Deng. Online feature selection for streaming features with high redundancy using sliding-window sampling. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 205–212. IEEE, 2018.

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

Appendix A. Detailed Proofs

A.1. Proof of Theorem 1

Proof By discarding the zero-component[6], the optimality conditions for objective (2) are formulated as follows.

$$\frac{1}{n}X_{\mathcal{A}}^T\widehat{\boldsymbol{e}(\theta)} + \alpha\nabla_{\boldsymbol{w}_{\mathcal{A}}}\mathcal{R}\left(\boldsymbol{w}\right) \overset{\text{def}}{=} \mathcal{F}\left(\theta, \boldsymbol{w}_{\mathcal{A}}, \boldsymbol{w}_{\mathcal{E}}\right) = \mathbf{0}, \tag{11}$$

$$\frac{\phi_\theta}{n}\cdot\widetilde{X}_{\mathcal{E}}^T\widehat{\boldsymbol{e}(\theta)} + \alpha\nabla_{\boldsymbol{w}_{\mathcal{E}}}\mathcal{R}\left(\boldsymbol{w}\right) \overset{\text{def}}{=} \mathcal{G}\left(\theta, \boldsymbol{w}_{\mathcal{A}}, \boldsymbol{w}_{\mathcal{E}}\right) = \mathbf{0}. \tag{12}$$

We derive a system of ODEs to compute the solution path w.r.t. θ when the active sets $\mathcal{A}$ and $\mathcal{E}$ are fixed. With the aid of derivation rules for compound function, we take the derivative of θ in (11) and (12). We can build

$$0 = \begin{cases} \dfrac{\partial\mathcal{F}(\cdot)}{\partial\theta} + \dfrac{\partial\mathcal{F}(\cdot)}{\partial\boldsymbol{w}_A}\cdot\dfrac{d\boldsymbol{w}_{\mathcal{A}}}{d\theta} + \dfrac{\partial\mathcal{F}(\cdot)}{\partial\boldsymbol{w}_{\mathcal{E}}}\cdot\dfrac{d\boldsymbol{w}_{\mathcal{E}}}{d\theta} \\ \dfrac{\partial\mathcal{G}(\cdot)}{\partial\theta} + \dfrac{\partial\mathcal{G}(\cdot)}{\partial\boldsymbol{w}_A}\cdot\dfrac{d\boldsymbol{w}_{\mathcal{A}}}{d\theta} + \dfrac{\partial\mathcal{G}(\cdot)}{\partial\boldsymbol{w}_{\mathcal{E}}}\cdot\dfrac{d\boldsymbol{w}_{\mathcal{E}}}{d\theta} \end{cases}. \tag{13}$$

The above two linear equations (13) can be converted to the following form:

$$\begin{aligned} \begin{bmatrix} \dfrac{\partial\mathcal{F}(\cdot)}{\partial\boldsymbol{w}_{\mathcal{A}}} & \dfrac{\partial\mathcal{F}(\cdot)}{\partial\boldsymbol{w}_{\mathcal{E}}} \\ \dfrac{\partial\mathcal{G}(\cdot)}{\partial\boldsymbol{w}_{\mathcal{A}}} & \dfrac{\partial\mathcal{G}(\cdot)}{\partial\boldsymbol{w}_{\mathcal{E}}} \end{bmatrix} \begin{bmatrix} \dfrac{d\boldsymbol{w}_{\mathcal{A}}}{d\theta} \\ \dfrac{d\boldsymbol{w}_{\mathcal{E}}}{d\theta} \end{bmatrix} &\overset{\text{def}}{=} \widehat{\mathcal{K}} \begin{bmatrix} \dfrac{d\boldsymbol{w}_{\mathcal{A}}}{d\theta} \\ \dfrac{d\boldsymbol{w}_{\mathcal{E}}}{d\theta} \end{bmatrix} \\ &= -\frac{1}{n}\begin{bmatrix} \phi_\theta'\cdot X_{\mathcal{A}}^T\widetilde{X}_{\mathcal{E}}\boldsymbol{w}_{\mathcal{E}} \\ \widetilde{X}_{\mathcal{E}}^T\left(\phi_\theta'\widehat{\boldsymbol{e}(\theta)} + \phi_\theta\phi_\theta'\widetilde{X}_{\mathcal{E}}\boldsymbol{w}_{\mathcal{E}}\right) \end{bmatrix}, \end{aligned} \tag{14}$$

where $(\cdot)'$ is the derivative note. By computing the remaining partial derivative terms in (14), we know that $\widehat{\mathcal{K}}$ has the form of $\begin{bmatrix} U & Z^T \\ Z & V \end{bmatrix}$, where $U = \frac{1}{n}X_{\mathcal{A}}^TX_{\mathcal{A}} + \alpha\nabla^2\mathcal{R}\left(\boldsymbol{w}_{\mathcal{A}}\right)$, $V = \frac{\phi_\theta^2}{n}\cdot\widetilde{X}_{\mathcal{E}}^T\widetilde{X}_{\mathcal{E}} + \alpha\nabla^2\mathcal{R}\left(\boldsymbol{w}_{\mathcal{E}}\right)$ and $Z = \frac{\phi_\theta}{n}\cdot\widetilde{X}_{\mathcal{E}}^TX_{\mathcal{A}}$. ■

A.2. Proof of Theorem 2

Proof Based on the symmetry property of any Hessian matrix $\nabla^2\mathcal{R}(\cdot) = \nabla(\nabla\mathcal{R}(\cdot))$, we can conclude $U^T = \frac{1}{n}(X_{\mathcal{A}}^TX_{\mathcal{A}})^T + \alpha(\nabla^2\mathcal{R})^T = U$. Meanwhile, the $V^T = V$ can be proved similarly. From the foregoing, we have $\widehat{\mathcal{K}}^T = \begin{bmatrix} U^T & Z^T \\ Z & V^T \end{bmatrix} = \begin{bmatrix} U & Z^T \\ Z & V \end{bmatrix} = \widehat{\mathcal{K}}$, which completes the proof. ■

6. Only pay attention to active sets $\mathcal{A}\cup\mathcal{E}$ and $\partial\mathcal{R}(\cdot)$ turns to the gradient of $\mathcal{R}(\cdot)$.

Appendix B. Practical Guides

This part we present the online incremental solver in detail for two significant learning models mentioned in the main body.

B.1. Group Lasso

At first, we compute the (sub)gradient of regularization term as

$$\partial \sum_{g \in G} \|\boldsymbol{w}^g\|_2 = \begin{cases} \frac{\boldsymbol{w}^g}{\|\boldsymbol{w}^g\|_2} & \text{if } \boldsymbol{w}^g \neq \mathbf{0} \\ \in \{u : \|u\|_2 \leqslant 1\} & \text{if } \boldsymbol{w}^g = \mathbf{0} \end{cases}. \tag{15}$$

There follows $\nabla^2 \mathcal{R}(\boldsymbol{w}_{\mathcal{A}\cup\mathcal{E}}) = diag(B_i)$, where $B_g = \frac{\sqrt{d_g}(\|\boldsymbol{w}^g\|^2 \mathcal{I} - \boldsymbol{w}^g \boldsymbol{w}^{g\top})}{\|\boldsymbol{w}^g\|^3}, g \in \mathcal{A} \cup \mathcal{E}$. $\mathcal{I}$ represents identity matrix.

Transition point for group Lasso First of all, we define the $\mathcal{C}_g$ as $\boldsymbol{X}_g^T \left(y - X_{\mathcal{A}} \boldsymbol{w}_{\mathcal{A}} - \theta \widetilde{X}_{\mathcal{E}} \boldsymbol{w}_{\mathcal{E}}\right)$, where $g = 1, \cdots, G$ denotes g-th predefined group. The KKT conditions for group Lasso problem are formulated as follows.

$$\begin{aligned} \mathcal{C}_g &= \frac{\alpha\sqrt{d_g} \cdot \boldsymbol{w}^g}{\|\boldsymbol{w}^g\|} \qquad \text{if } g \in \mathcal{A}, \\ \theta\mathcal{C}_g &= \frac{\alpha\sqrt{d_g} \cdot \boldsymbol{w}^g}{\|\boldsymbol{w}^g\|} \qquad \text{if } g \in \mathcal{E}. \end{aligned} \tag{16}$$

A non-zero $\boldsymbol{w}^g$ will decay to $\mathbf{0}$ as $\|\mathcal{C}_g\| \leqslant \alpha\sqrt{d_g}, g \in \mathcal{A}$ (or $\theta \|\mathcal{C}_g\| \leqslant \alpha\sqrt{d_g}, g \in \mathcal{E}$) is found. Based on the continuity of solutions given by ODEs solver, we can set $\boldsymbol{w}^g = \mathbf{0}$ $(g \in \mathcal{A} \cup \mathcal{E})$ as soon as it reaches the opposite sign (i.e. path pass through 0), meanwhile, index g is removed from set $\mathcal{A}$.

Alternatively, when the norm value of $\mathcal{C}_g, g \in \overline{\mathcal{A}}$ (or $\phi_\theta \|\mathcal{C}_g\|, g \in \mathcal{E}$) comes to $\alpha\sqrt{d_g}$, the g-th group becomes active and will be added into active set. (16) indicates that $\mathcal{C}_g (g \in \mathcal{A} \cup \mathcal{E})$ shares the collinearity with $\boldsymbol{w}^g$. Therefore an extreme short syntropic coefficient vector $\epsilon\mathcal{C}_g$ may be used as a 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 $\overline{\mathcal{A}}$ (or $\theta\|\mathcal{C}_g\| - \alpha\sqrt{d_g}$ for each g in $\overline{\mathcal{E}}$) while solving the solution path to estimate transition point.

B.2. ℓ_p Penalization

We give a start value of $\boldsymbol{w}_2$ at $\theta = 0$ by optimal conditions. Detailed steps are shown in our Algorithm 2.

Algorithm 1 `DFLS` on Group Lasso (**incremental**)

Require: Samples $\mathbf{X} = \left[X, \widetilde{X}\right]$, labels $\boldsymbol{y}$, initial solution $\boldsymbol{w}|_{\theta=0}$, regularization strength α.

Set $\mathcal{A}$, $\overline{\mathcal{A}}$ and $\mathcal{E}$, $\overline{\mathcal{E}}$ according to $\boldsymbol{w}|_{\theta=0}$.

Compute the first transition point $\widetilde{\theta} = \min \left\{ \dfrac{\alpha n \sqrt{d_g}}{\left\| \widetilde{X}_g^T \left(y - X_{\mathcal{A}} \boldsymbol{w}_{\mathcal{A}} \right) \right\|} \right\}$, for $g \in \overline{\mathcal{E}}$.

Compute clipped $\widehat{\theta} = \min \left\{ \widetilde{\theta}, 1 \right\}$.

Update $\boldsymbol{w}|_{\theta=\widehat{\theta}} = \boldsymbol{w}|_{\theta=0}$.

if $\widehat{\theta} < 1$ **then**

  while $\theta \leq 1$ **do**

    Solve (8) and detect transition point simultaneously.

    if g-th group turns to inactive **then**

      $\boldsymbol{w}_g = \mathbf{0}$.

      Move g from $\mathcal{A}$ (or $\mathcal{E}$) to $\overline{\mathcal{A}}$ (or $\overline{\mathcal{E}}$).

    else if g-th group becomes active **then**

      $\boldsymbol{w}_g = \epsilon \mathcal{C}_g$.

      Put g into $\mathcal{A}$ (or $\mathcal{E}$).

    end if

    Update $\boldsymbol{w}_{\mathcal{A}}$, $\boldsymbol{w}_{\mathcal{E}}$, $X_{\mathcal{A}}$ and $\widetilde{X}_{\mathcal{E}}$ according to the updated $\mathcal{A}$ and $\mathcal{E}$.

  end while

end if

Ensure: $\boldsymbol{w}|_{\theta=1}$

Algorithm 2 `DFLS` on ℓ_p Penalization (**incremental**)

Require: Samples $\mathbf{X} = \left[X, \widetilde{X}\right]$, labels $\boldsymbol{y}$, initial solution $\boldsymbol{w}|_{\theta=0}$, regularization strength α, parameter $p > 1$.

Set $\boldsymbol{w}_2|_{\theta=0}$ by optimal conditions.

while $\theta \leq 1$ **do**

  Solve (8).

end while

Ensure: $\boldsymbol{w}|_{\theta=1}$

Appendix C. Path Visualization

We plot the solution path of two types of regularizers in this section, which helps readers better understand their property. Figure 2 shows an example of solution path of group Lasso with respect to successively varied θ, where $\theta = 1$ corresponds to the adjustment of incorporating new features and $\theta = 0$ corresponds to removing these features.

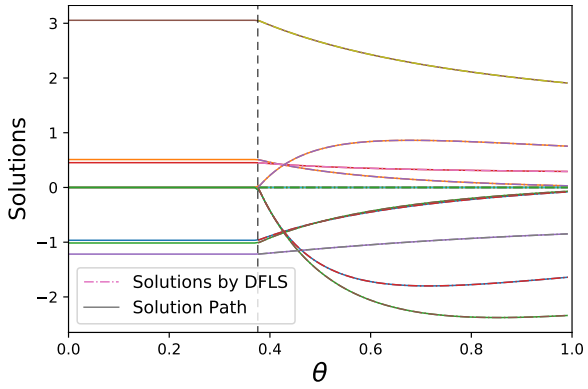


Figure 2: Piecewise smooth path of θ when adjusting features in group Lasso, where one piecewise smooth curve corresponds to one variable of $\boldsymbol{w}$. The dotted lines represent solutions given by `DFLS`, which coincide with the real solution path (in solid lines).

The ℓ_p penalization does not bring sparsity, so there is no transition point. The figure 3 shows an example of its path visualization.

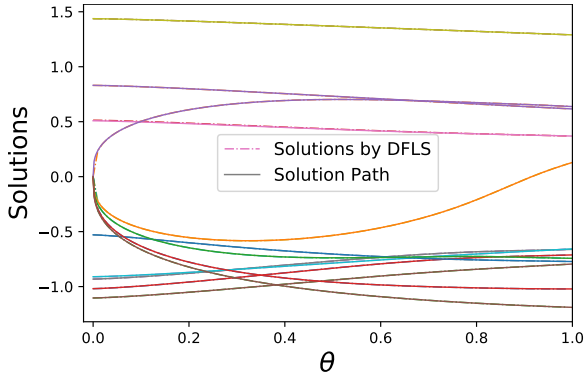


Figure 3: Piecewise smooth path of θ when adjusting features in ℓ_4 penalization. The dotted lines represent solutions given by `DFLS`, which coincide with the real solution path.

Appendix D. Implementation Details.

We implement our novel algorithms in Python 3.7. 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 [15], 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. The standard deviation of the Gaussian centered noise applied to the output is 1. This setting is common in lots of related works. The stop criterion of the batch training algorithm is set to 1e-7.[7]

More importantly, we will provide our *directly runnable* code with a "Readme" tutorial in Supplementary Material, interested researchers can run Python 3.7 code to reproduce the main results.

Appendix E. Feature Decremental Learning

E.1. Group Lasso

We present the feature decremental learning procedure of group Lasso in Algorithm 3.

To reduce dimensionality, we can track the solution path as θ varies from 1 to 0, which can be considered as an inverse process of incremental learning. In particular, we can directly eliminate $\widetilde{X}$ as we found $\mathcal{E} = \varnothing$ (*i.e.*, they are irrelevant or redundant features). Similar to incremental learning for features, this is supported by equations (4) & (5) of main body. At last, we set $\boldsymbol{w}_{\mathcal{E}\cup\overline{\mathcal{E}}}$ to exact 0 to complete the whole process.

Algorithm 3 `DFLS` on Group Lasso (**decremental**)

Require: Samples $\boldsymbol{X} = \left[X, \widetilde{X}\right]$, labels $\boldsymbol{y}$, initial solution $\boldsymbol{w}|_{\theta=1}$, regularization strength α.

Set $\mathcal{A}$, $\overline{\mathcal{A}}$ and $\mathcal{E}$, $\overline{\mathcal{E}}$ according to $\boldsymbol{w}|_{\theta=1}$.
while $\mathcal{E} \neq \varnothing$ and $\theta > 0$ **do**
  Solve $\dfrac{d\boldsymbol{w}_{\mathcal{A}\cup\mathcal{E}}}{d\theta} = \begin{bmatrix} \beta_{\mathcal{A}} \\ \beta_{\mathcal{E}} \end{bmatrix}$ and detect transition point simultaneously.
  if g-th group turns to inactive **then**
    $\boldsymbol{w}_g = \mathbf{0}$.
    Move g from $\mathcal{A}$ (or $\mathcal{E}$) to $\overline{\mathcal{A}}$ (or $\overline{\mathcal{E}}$).
  else if g-th group becomes active **then**
    $\boldsymbol{w}_g = \epsilon \mathcal{C}_g$.
    Put g into $\mathcal{A}$ (or $\mathcal{E}$).
  end if
  Update $\boldsymbol{w}_{\mathcal{A}}$, $\boldsymbol{w}_{\mathcal{E}}$, $X_{\mathcal{A}}$ and $\widetilde{X}_{\mathcal{E}}$ according to the updated $\mathcal{A}$ and $\mathcal{E}$.
end while
Set $\boldsymbol{w}_{\mathcal{E}\cup\overline{\mathcal{E}}} = \mathbf{0}$.

Ensure: $\boldsymbol{w}|_{\theta=0}$

E.2. ℓ_p-norm Regularized Regression

At this part, we present the pseudo-code of feature decremental algorithm for ℓ_p-norm regularized regression in Algorithm 4.

7. Besides, the accuracy results reveal that the solution of our approach has the same quality as that of the batch method.

Algorithm 4 DFLS on ℓ_p Penalization (**decremental**)

Require: Samples $\boldsymbol{X} = \left[X, \widetilde{X}\right]$, labels $\boldsymbol{y}$, initial solution $\boldsymbol{w}|_{\theta=1}$, regularization strength α, parameter $p > 1$.

1: **while** $\theta > 0$ **do**
2: Solve ODEs to compute the solution path.
3: **end while**
4: Set $\boldsymbol{w}_{\mathcal{E}\cup\overline{\mathcal{E}}} = \mathbf{0}$.

Ensure: $\boldsymbol{w}|_{\theta=0}$

E.3. Simulation Studies

In this subsection, we empirically show the accuracy of derived ODEs in Table 5 and Table 6. In experimental settings, DFLS on different group partitions in group Lasso (denotes as G) and various ℓ_p-norm are used to perform features shrinkage while different regularization parameter α are chosen. We conduct comparative experiments in the same environment as in the main paper.

Dataset		Yolanda		BNG(libras_move)		satellite_image		BNG(wisconsin)		cpu_act	
G	α	DFLS	batch	DFLS	batch	DFLS	batch	DFLS	batch	DFLS	batch
I	0.2	10.64	10.65	4.10	4.10	1.54	1.54	29.97	29.97	10.05	10.05
	0.4	10.78	10.78	4.24	4.24	1.77	1.76	30.54	30.54	10.17	10.18
II	0.2	10.74	10.74	4.08	4.09	1.57	1.57	30.04	30.04	9.97	9.98
	0.4	10.89	10.90	4.32	4.32	1.82	1.82	30.43	30.42	10.14	10.14

Table 5: Results of aRMSE when training the group Lasso with dynamic features (**decremental**). Any variance less than 10^{-5} are omitted. DFLS and *batch* represent each iteration is trained with a chunk of new features, or using batch algorithm to retrain from scratch, respectively.

Dataset		Yolanda		BNG(libras_move)		satellite_image		BNG(wisconsin)		cpu_act	
ℓ_p	α	DFLS	batch	DFLS	batch	DFLS	batch	DFLS	batch	DFLS	batch
ℓ_4	0.2	9.83	9.83	3.76	3.76	1.21	1.21	31.41	31.41	15.38	15.38
	0.5	9.90	9.90	3.77	3.77	1.21	1.21	31.83	31.83	15.81	15.81
ℓ_6	0.2	9.84	9.84	3.76	3.76	1.20	1.21	31.77	31.77	15.84	15.84
	0.5	9.88	9.88	3.76	3.78	1.20	1.21	32.05	32.05	16.07	16.07

Table 6: The aRMSE results of ℓ_p-norm regularized regression model (**decremental**). Any variance less than 10^{-5} are omitted.

Appendix F. Additional Discussion

This section we will give some discussions on our DFLS.

Note if the whole path is smooth on $(0, 1)$ (*i.e.,* when $\widetilde{\theta} \geq 1$ for sparsity-based penalization or non-sparsity regularization $\mathcal{R}(\cdot)$ is used), algorithm needn't cost extra time on detecting specific locations and recalculating the solution path, which apparently reduces the computation complexity.

The concrete process of designing an algorithm for dynamic updating by `DFLS` mainly consists of computing $\nabla^2\mathcal{R}(\cdot)$ and investigating optimality conditions. Hence, from a computational perspective, regularizer with easily computed $\nabla^2\mathcal{R}(\cdot)$ and $\nabla\mathcal{R}(\cdot)$ can have a better adaptation to our `DFLS` framework.