Efficient Adaptive Online Learning via Frequent Directions

Yuanyu Wan and Lijun Zhang, *Member, IEEE*

Abstract—By employing time-varying proximal functions, adaptive subgradient methods (ADAGRAD) have improved the regret bound and been widely used in online learning and optimization. However, ADAGRAD with full matrix proximal functions (ADA-FULL) cannot handle large-scale problems due to the impractical $O(d^3)$ time and $O(d^2)$ space complexities, though it has better performance when gradients are correlated. In this paper, we propose two efficient variants of ADA-FULL via a matrix sketching technique called frequent directions (FD). The first variant named as ADA-FD directly utilizes FD to maintain and manipulate low-rank matrices, which reduces the space and time complexities to $O(\tau d)$ and $O(\tau^2 d)$ respectively, where d is the dimensionality and $\tau \ll d$ is the sketching size. The second variant named as ADA-FFD further adopts a doubling trick to accelerate FD used in ADA-FD, which reduces the average time complexity to $O(\tau d)$ while only doubles the space complexity of ADA-FD. Theoretical analysis reveals that the regret of ADA-FD and ADA-FFD is close to that of ADA-FULL as long as the outer product matrix of gradients is approximately low-rank. Experimental results demonstrate the efficiency and effectiveness of our algorithms.

Index Terms—Online learning, frequent directions, adaptive subgradient methods

1 Introduction

Online learning refers to the process of answering a sequence of questions given answers to previous questions, which enjoys an attractive combination of computational efficiency and theoretical guarantees [2]. Recently, it has received ever-increasing attention due to the emergence of large-scale applications such as online classification [3], [4], [5], [6], [7], online advertisements recommendation [8], [9], [10], online metric learning [11], [12], [13], [14], online matrix factorization [15], [16], [17], [18], online optimization [19], [20], [21], [22], [23], and online anomaly detection [24], [25]. Adaptive subgradient methods (ADAGRAD), which employ proximal functions that adapt to the geometry of the data observed in earlier iterations, are popular for online learning and optimization [26]. According to the type of proximal functions, ADAGRAD can be divided into learning with full matrix proximal functions (ADA-FULL) and learning with diagonal matrix proximal functions (ADA-DIAG). In contrast to ADA-FULL, ADA-DIAG has been successfully applied to many large-scale applications because of its light computations and storages.

However, the diagonal matrix maintained in ADA-DIAG only contains limited information of gradient outer products. This shortcoming causes that the regret of ADA-DIAG is worse than that of ADA-FULL when the high-dimensional data are dense and have an approximately low-rank structure. In consideration of this dilemma, Duchi *et al.* [26] proposed an open question concerning whether we can efficiently use full matrices in the proximal functions. To solve this problem, Krummenacher *et al.* [27] utilized random projections to approximate ADA-FULL, and developed two methods, namely ADA-LR and RADAGRAD. Compared with ADA-FULL, ADA-LR has the same space complexity $O(d^2)$ and a slightly reduced time complexity $O(\tau d^2)$, where d is the dimensionality and $\tau \ll d$ is the number of random projections. Due to the quadratic dependence on d, ADA-LR is impractical for high-dimensional data, though it is equipped with a formal regret bound. In contrast, the time and space complexities of RADAGRAD are linear in d, but it lacks theoretical guarantees owing to further approximations. In our recent work [28], we also utilized random projections to accelerate ADA-FULL, and proposed ADA-DP that has complexities linear in d. However, its theoretical guarantees are non-deterministic, and the regret bound contains additional terms that never vanish.

To tackle the above limitations, we develop two efficient variants of ADA-FULL which are also principled. The computational bottleneck of ADA-FULL is to store the outer product matrix of gradients and compute its square root in each round. We note that a similar bottleneck also exists in other second order methods such as online Newton step (ONS) [29], and recently matrix sketching techniques have been used to reduce the computational cost of ONS [30]. Motivated by previous work, we first propose to directly employ frequent directions [31] to approximate the outer product matrix of gradients in ADA-FULL. By utilizing the low-rank structure, our first efficient variant of ADA-FULL, named ADA-FD, reduces the space complexity from $O(d^2)$ to $O(\tau d)$ and the time complexity from $O(d^3)$ to $O(\tau^2 d)$, where τ is the sketching size, which implies both the space and time complexities of ADA-FD are linear in the dimensionality d. Then, we further propose a much faster variant of ADA-FULL, named ADA-FFD, by accelerating frequent directions used in ADA-FD, which

• *The authors are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, China. E-mail: {wanyy, zhanglj}@lamda.nju.edu.cn.*

*Manuscript received 26 Apr. 2019; revised 24 May 2021; accepted 7 July 2021. Date of publication 14 July 2021; date of current version 9 Sept. 2022.
(Corresponding author: Lijun Zhang.)
Recommended for acceptance by A. Gyorgy.
Digital Object Identifier no. 10.1109/TPAMI.2021.3096880*

only needs to double the space complexity. Assuming $\tau \leq \sqrt{d}$, ADA-FFD reduces the average time complexity to $O(\tau d)$ that is linear in both the dimensionality d and the sketching size τ.

Moreover, because the idea of ADAGRAD can be incorporated into either the primal-dual subgradient method [32] or the composite mirror descent method [33], we also develop two efficient methods for both ADA-FD and ADA-FFD according to the type of subgradient methods, and prove that the regret bounds of our methods are close to that of ADA-FULL. The slight differences in the regret bounds are caused by the approximation error of frequent directions or fast frequent directions, and vanish when the sketching size is larger than the rank of gradient outer products. More generally, when the outer product matrix of gradients is approximately low-rank, a small number of sketching is sufficient to prevent the differences from affecting the order of the regret bounds. To verify the efficiency and effectiveness of our ADA-FD and ADA-FFD, we conduct numerical experiments on online regression, online classification, training convolutional neural networks (CNN), and fine-tuning much deeper CNN. The results turn out that our ADA-FD and ADA-FFD perform comparably with ADA-FULL, but are much more efficient.

2 Related Work

In this section, we briefly review the related work in adaptive subgradient methods and matrix sketching.

2.1 ADAGRAD

Adaptive subgradient methods describe and analyze an apparatus for learning the proximal functions which are modified according to data observed in earlier iterations. This property significantly simplifies choosing the step size while ensures regret guarantees as good as the proximal functions tuned manually. In the following, we provide a brief introduction of ADAGRAD for the primal-dual subgradient method [32] and the composite mirror descent method [33].

At each round $t = 1, 2, \ldots, T$, the online learner predicts a point $\boldsymbol{\beta}_t \in \mathbb{R}^d$, and then the adversary reveals a composite function $F_t(\boldsymbol{\beta}) = f_t(\boldsymbol{\beta}) + \varphi(\boldsymbol{\beta})$, where f_t and φ are convex. The learner suffers a loss $F_t(\boldsymbol{\beta}_t)$ for this round, and the goal of the learner is to minimize the regret that is defined as

$$R(T) = \sum_{t=1}^{T} F_t(\boldsymbol{\beta}_t) - \sum_{t=1}^{T} F_t(\boldsymbol{\beta}^*),$$

where $\boldsymbol{\beta}^*$ is the fixed optimal predictor. Let $\mathbf{g}_t$ be a particular vector in the subdifferential set $\partial f_t(\boldsymbol{\beta}_t)$ of the function f_t. The outer product matrix of gradients is defined as $\mathrm{G}_t = \sum_{i=1}^{t} \mathbf{g}_i \mathbf{g}_i^\top$, and we further define a symmetric matrix H_t, which has the following two choices:

$$\mathrm{H}_t = \begin{cases} \delta \mathrm{I}_d + \mathrm{diag}(\mathrm{G}_t)^{1/2} & \text{ADA-DIAG} \\ \delta \mathrm{I}_d + \mathrm{G}_t^{1/2} & \text{ADA-FULL} \end{cases},$$

where $\delta > 0$ is a parameter making H_t invertible. The proximal term is given by $\Psi_t(\boldsymbol{\beta}) = \frac{1}{2}\langle \boldsymbol{\beta}, \mathrm{H}_t \boldsymbol{\beta}\rangle$, and let

$$B_{\Psi_t}(\mathbf{x}, \mathbf{y}) = \Psi_t(\mathbf{x}) - \Psi_t(\mathbf{y}) - \langle \nabla \Psi_t(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle$$

denote the Bregman divergence associated with Ψ_t. Let $\eta > 0$ be a fixed step size and $\bar{\mathbf{g}}_t = \sum_{i=1}^{t} \mathbf{g}_i$. In each iteration, the primal-dual subgradient method updates by

$$\begin{aligned} \boldsymbol{\beta}_{t+1} &= \arg\min_{\boldsymbol{\beta}} \left\{ \eta \left\langle \frac{1}{t} \bar{\mathbf{g}}_t, \boldsymbol{\beta} \right\rangle + \eta \varphi(\boldsymbol{\beta}) + \frac{1}{t} \Psi_t(\boldsymbol{\beta}) \right\} \\ &= -\eta \mathrm{H}_t^{-1} \bar{\mathbf{g}}_t, \quad \text{if } \varphi = 0. \end{aligned} \tag{1}$$

And the composite mirror descent method updates by

$$\begin{aligned} \boldsymbol{\beta}_{t+1} &= \arg\min_{\boldsymbol{\beta}} \{ \eta \langle \mathbf{g}_t, \boldsymbol{\beta} \rangle + \eta \varphi(\boldsymbol{\beta}) + B_{\Psi_t}(\boldsymbol{\beta}, \boldsymbol{\beta}_t) \} \\ &= \boldsymbol{\beta}_t - \eta \mathrm{H}_t^{-1} \mathbf{g}_t, \quad \text{if } \varphi = 0. \end{aligned} \tag{2}$$

Because the storage complexity of G_t is $O(d^2)$ and the time complexity of finding its square root is $O(d^3)$, ADA-FULL is impractical for large-scale problems.

To reduce the complexities of ADA-FULL, Krummenacher *et al.* [27] proposed two methods based on the fast randomized singular value decomposition (SVD) [34] to approximate the proximal term $\Psi_t(\boldsymbol{\beta})$. Let $\Pi \in \mathbb{R}^{\tau \times d}$ be the random matrix of the subsampled randomized Fourier transform. They first used the following steps to update $\boldsymbol{\beta}_t$:

$$\begin{aligned} &\mathrm{G}_t = \mathrm{G}_{t-1} + \mathbf{g}_t \mathbf{g}_t^\top \\ &\widetilde{\mathrm{G}}_t = \mathrm{G}_t \Pi^\top \qquad \text{Random Projection} \\ &\mathrm{QR} = \widetilde{\mathrm{G}}_t \qquad \text{QR-decomposition} \\ &\mathrm{B} = \mathrm{Q}^\top \mathrm{G}_t, \mathrm{U\Sigma V}^\top = \mathrm{B} \qquad \text{SVD} \\ &\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta \mathrm{V}(\Sigma^{1/2} + \delta \mathrm{I}_\tau)^{-1} \mathrm{V}^\top \mathbf{g}_t, \end{aligned} \tag{3}$$

and the resulting algorithm is refereed to as ADA-LR. Note that the space and time complexities of ADA-LR are respectively $O(d^2)$ and $O(\tau d^2)$, which prevent ADA-LR from being practical for high-dimensional data. By introducing more randomized approximations, they presented a more scalable algorithm RADAGRAD that performs the following updates:

$$\begin{aligned} &\widetilde{\mathbf{g}}_t = \Pi \mathbf{g}_t \qquad \text{Random Projection} \\ &\widetilde{\mathrm{G}}_t = \widetilde{\mathrm{G}}_{t-1} + \mathbf{g}_t \widetilde{\mathbf{g}}_t^\top \\ &[\mathrm{Q}_t, \mathrm{R}_t] = \text{qr\_update}(\mathrm{Q}_{t-1}, \mathrm{R}_{t-1}, \mathbf{g}_t, \widetilde{\mathbf{g}}_t) \\ &\mathrm{B} = \widetilde{\mathrm{G}}_t^\top \mathrm{Q}_t, \mathrm{U\Sigma W}^\top = \mathrm{B} \qquad \text{SVD} \\ &\mathrm{V} = \mathrm{Q}_t \mathrm{W}, \gamma_t = \eta(\mathbf{g}_t - \mathrm{VV}^\top \mathbf{g}_t) \\ &\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta \mathrm{V}(\Sigma^{1/2} + \delta \mathrm{I}_\tau)^{-1} \mathrm{V}^\top \mathbf{g}_t - \gamma_t, \end{aligned} \tag{4}$$

where qr_update means QR decomposition with the rank-one update. In this case, RADAGRAD reduces the space and time complexities to $O(\tau d)$ and $O(\tau^2 d)$ respectively. Unfortunately, it is a heuristic method and lacks theoretical guarantees. When $f_t(\boldsymbol{\beta}_t) = l(\boldsymbol{\beta}_t^\top \mathbf{x}_t)$ where $\mathbf{x}_t$ is a data vector independent from the learning algorithm, we recently proposed ADA-DP [28] based on random projections to attain theoretical guarantees and keep complexities linear in d. However, due to the randomness of random projections, the regret bound of ADA-DP is non-deterministic and is worse than ADA-FULL even when $\tau = d$.

2.2 Matrix Sketching and Frequent Directions

For any given matrix $\mathrm{A} \in \mathbb{R}^{t\times d}$, the purpose of sketching is to generate a matrix $\mathrm{B} \in \mathbb{R}^{\tau\times d}$ where $\tau \ll t$ is the sketching size such that $\mathrm{A} \approx \mathrm{B}$ or $\mathrm{A}^\top\mathrm{A} \approx \mathrm{B}^\top\mathrm{B}$. There are many matrix sketching techniques including frequent directions [31], [35], [36], random projection [37], [38], [39], and column selection [40]. Frequent directions [31] is a deterministic matrix sketching technique by extending the well-known algorithm for approximating item frequencies in streams [41]. Let $\mathrm{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_\tau]^\top = \mathbf{0}_{\tau\times d}$ where $\tau \ll \min\{t, d\}$. For any given matrix $\mathrm{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_t]^\top$, frequent directions processes each row of A as follows:

$$\begin{aligned}&\mathbf{b}_\tau = \mathbf{a}_i, \mathrm{U}\Sigma\mathrm{V}^\top = \mathrm{B} \qquad \text{SVD}\\ &\mathrm{B} = \sqrt{\Sigma^2 - \sigma\mathrm{I}_\tau}\mathrm{V}^\top \text{ where } \sigma = \Sigma^2_{\tau\tau},\end{aligned}$$

and products a sketch matrix B. Because the time complexity of processing each row is $O(\tau^2 d)$ that is dominated by computing the SVD of B, the total time complexity of frequent directions is $O(\tau^2 dt)$ which suffers quadratic dependence on the sketching size τ. To further reduce it to $O(\tau dt)$, Ghashami *et al.* [31] have proposed fast frequent directions (FFD) at the expense of doubling the space complexity. Let $\mathrm{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{2\tau}]^\top = \mathbf{0}_{2\tau\times d}$. Fast frequent directions processes each row of A as follows:

```
Insert $\mathbf{a}_i$ into a zero valued row of B
if B has no zero valued rows then
    $\mathrm{U}\Sigma\mathrm{V}^\top = \mathrm{B}$        SVD
    $\mathrm{B} = \sqrt{\max(\Sigma^2 - \sigma\mathrm{I}_{2\tau}, 0)}\mathrm{V}^\top$ where $\sigma = \Sigma^2_{\tau\tau}$        (5)
endif,
```

where the *if* statement is only triggered once every $\tau + 1$ rows. Therefore, the number of computing the SVD of B is only $t/(\tau + 1)$ and the total time complexity of fast frequent directions is $O(\tau^2 d(t/\tau)) = O(\tau dt)$.

Recently, the techniques of matrix sketching have been used by Luo *et al.* [30] to accelerate online Newton step [29]. They studied ONS that updates by

$$\mathrm{A}_t = \alpha\mathrm{I}_d + \sum_{i=1}^t \eta_i\mathbf{g}_i\mathbf{g}_i^\top \text{ and } \boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \mathrm{A}_t^{-1}\mathbf{g}_t, \tag{6}$$

where $\alpha > 0$ and $\eta_t > 0$ are some parameters for general convex functions, and used matrix sketching techniques to construct a low-rank approximation of the second order information. Motivated by Luo *et al.* [30], our work employs frequent directions to calculate a low-rank approximation of the full matrix, which obviously reduces the storage and time complexities of ADA-FULL. Compared to the approximation algorithm used by Krummenacher *et al.* [27], frequent directions has two advantages which are deterministic theoretical properties and easy implementations.

3 Efficient Variants of ADA-FULL

In this section, we first introduce our two efficient variants of ADA-FULL and the corresponding theoretical results. Then, we compare our work with Krummenacher *et al.* [27] and Luo *et al.* [30]. To facilitate presentations, we consider the case $\varphi = 0$, and our methods can be extended to the general case $\varphi \neq 0$.

3.1 Adaptive Online Learning via FD

Define $\mathrm{C}_t = [\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_t]^\top \in \mathbb{R}^{t\times d}$, where each row is a gradient vector. The outer product matrix of gradients can be calculated as

$$\mathrm{G}_t = \sum_{i=1}^t \mathbf{g}_i\mathbf{g}_i^\top = \mathrm{C}_t^\top\mathrm{C}_t.$$

To accelerate the computation of H_t^{-1}, we take advantage of frequent directions to produce a low-rank approximation of C_t denoted by $\mathrm{S}_t = [\mathbf{s}_1^t, \mathbf{s}_2^t, \dots, \mathbf{s}_\tau^t]^\top \in \mathbb{R}^{\tau\times d}$. Then we can redefine H_t in the proximal term as

$$\mathrm{H}_t = \delta\mathrm{I}_d + (\mathrm{S}_t^\top\mathrm{S}_t)^{1/2}.$$

Let the SVD of S_t be $\mathrm{S}_t = \mathrm{U}\Sigma\mathrm{V}^\top$ where $\mathrm{U} \in \mathbb{R}^{\tau\times\tau}, \Sigma \in \mathbb{R}^{\tau\times\tau}$ and $\mathrm{V} \in \mathbb{R}^{d\times\tau}$, then we have $(\mathrm{S}_t^\top\mathrm{S}_t)^{1/2} = \mathrm{V}\Sigma\mathrm{V}^\top$. By Woodbury formula [42], we have

$$\begin{aligned}\mathrm{H}_t^{-1} =&(\delta\mathrm{I}_d + \mathrm{V}\Sigma\mathrm{V}^\top)^{-1}\\ =&\frac{1}{\delta}\left(\mathrm{I}_\mathrm{d} - \mathrm{V}(\delta\mathrm{I}_\tau + \Sigma)^{-1}\Sigma\mathrm{V}^\top\right).\end{aligned}$$

With the above procedures, we propose our first efficient variant of ADA-FULL, named as adaptive online learning via frequent directions (ADA-FD). Then, we first consider incorporating ADA-FD into the primal-dual subgradient method. According to the update rules in (1), in the tth round, our algorithm performs the following updates:

$$\begin{aligned}&\mathbf{s}_\tau^{t-1} = \mathbf{g}_t, \bar{\mathbf{g}}_t = \bar{\mathbf{g}}_{t-1} + \mathbf{g}_t\\ &\mathrm{U}\Sigma\mathrm{V}^\top = \mathrm{S}_{t-1}, \sigma_t = \Sigma^2_{\tau\tau} \qquad \text{SVD}\\ &\Sigma' = \sqrt{\Sigma^2 - \sigma_t\mathrm{I}_\tau}, \mathrm{S}_t = \Sigma'\mathrm{V}^\top\\ &\boldsymbol{\beta}_{t+1} = -\frac{\eta}{\delta}\left(\bar{\mathbf{g}}_t - \mathrm{V}(\delta\mathrm{I}_\tau + \Sigma')^{-1}\Sigma'\mathrm{V}^\top\bar{\mathbf{g}}_t\right).\end{aligned} \tag{7}$$

The detailed procedures of ADA-FD for the primal-dual subgradient method are summarized in Algorithm 1, and this method is named as adaptive dual averaging via frequent directions.

Moreover, we incorporate ADA-FD into the composite mirror descent method. According to the update rules in (2), in the tth round, our algorithm performs the following updates:

$$\begin{aligned}&\mathbf{s}_\tau^{t-1} = \mathbf{g}_t\\ &\mathrm{U}\Sigma\mathrm{V}^\top = \mathrm{S}_{t-1}, \sigma_t = \Sigma^2_{\tau\tau} \qquad \text{SVD}\\ &\Sigma' = \sqrt{\Sigma^2 - \sigma_t\mathrm{I}_\tau}, \mathrm{S}_t = \Sigma'\mathrm{V}^\top\\ &\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \frac{\eta}{\delta}\left(\mathbf{g}_t - \mathrm{V}(\delta\mathrm{I}_\tau + \Sigma')^{-1}\Sigma'\mathrm{V}^\top\mathbf{g}_t\right).\end{aligned} \tag{8}$$

The detailed procedures of ADA-FD for the composite mirror descent method are summarized in Algorithm 2, and this method is named as adaptive mirror descent via frequent directions.

3.1.1 Computational Complexity

First, the space complexities of our two methods are $O(\tau d)$ caused by the maintenance of $\mathrm{S}_t, \mathrm{U}, \mathrm{V}$. And the time complexities of our two methods are $O(\tau^2 d)$ caused by step 5 in each algorithm which contains the SVD. Thus, both of them are linear in the dimensionality d. Second, compared with the update rules of ADA-LR (3) and RADAGRAD (4), our update rules (7) and (8) do not need random projection and are more simple. Third, when $\varphi \neq 0$, the computational cost of H_t^{-1} can still be reduced dramatically, though the updating of $\boldsymbol{\beta}_t$ may not have a closed-form solution.

Algorithm 1. Adaptive Dual Averaging via FD

```
1: Input: $\eta > 0, \delta > 0, \tau, \mathrm{S}_0 = \mathbf{0}_{\tau\times d}, \bar{\mathbf{g}}_0 = \mathbf{0}, \boldsymbol{\beta}_1 = \mathbf{0}$
2: for $t = 1, \ldots, T$ do
3:     Receive the gradient $\mathbf{g}_t = \nabla f_t(\boldsymbol{\beta}_t)$
4:     Insert $\mathbf{g}_t$ into the last row of $\mathrm{S}_{t-1}$ and $\bar{\mathbf{g}}_t = \bar{\mathbf{g}}_{t-1} + \mathbf{g}_t$
5:     Compute $\mathrm{U\Sigma V}^\top = \mathrm{S}_{t-1}$, $\sigma_t = \Sigma^2_{\tau\tau}$
6:     Compute $\mathrm{S}_t = \Sigma'\mathrm{V}^\top$ where $\Sigma' = \sqrt{\Sigma^2 - \sigma_t \mathrm{I}_\tau}$
7:     $\boldsymbol{\beta}_{t+1} = -\frac{\eta}{\delta}\left(\bar{\mathbf{g}}_t - \mathrm{V}(\delta\mathrm{I}_\tau + \Sigma')^{-1}\Sigma'\mathrm{V}^\top\bar{\mathbf{g}}_t\right)$
8: end for
```

3.1.2 Theoretical Guarantees

Compared with RADAGRAD, a significant advantage of our methods is that they are equipped with formal theoretical guarantees similar to ADA-FULL. We present regret bounds for Algorithms 1 and 2 in the following two theorems, respectively.

Theorem 1. *Adaptive dual averaging via frequent directions ensures*

$$\begin{aligned}R(T) \le & \frac{\delta}{2\eta}\|\boldsymbol{\beta}^*\|_2^2 + \frac{1}{2\eta}\mathrm{tr}(\mathrm{G}_T^{1/2})\|\boldsymbol{\beta}^*\|_2^2 \\ & + \eta\max\left(1, \frac{\max_{t\le T}\|\mathbf{g}_t\|_2 + \sqrt{\Delta_T}}{\delta}\right)\mathrm{tr}(\mathrm{G}_T^{1/2}) \\ & + \frac{\sum_{t=1}^T\sqrt{\sigma_t}}{2\eta}\max_{t\le T}\|\boldsymbol{\beta}_{t+1}\|_2^2,\end{aligned} \tag{9}$$

where $\Delta_T = \sum_{i=1}^T \sigma_i$.

Theorem 2. *Adaptive mirror descent via frequent directions ensures*

$$\begin{aligned}R(T) \le & \frac{\delta}{2\eta}\|\boldsymbol{\beta}_*\|_2^2 + \frac{1}{2\eta}\max_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2\mathrm{tr}(\mathrm{G}_T^{1/2}) \\ & + \eta\max\left(1, \frac{\sqrt{\Delta_T}}{\delta}\right)\mathrm{tr}(\mathrm{G}_T^{1/2}) \\ & + \frac{\tau\sum_{t=1}^T\sqrt{\sigma_t}}{2\eta}\max_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2,\end{aligned} \tag{10}$$

where $\Delta_T = \sum_{i=1}^T \sigma_i$.

For comparisons, let us introduce the theoretical guarantees of ADA-FULL as follows.

Theorem 3. *(Theorem 7 of Duchi* et al. *[26]) Provided with* $\delta \ge \max_{t\le T}\|\mathbf{g}_t\|_2$*, ADA-FULL incorporated into the primal-dual subgradient method ensures*

$$R(T) \le \frac{\delta}{2\eta}\|\boldsymbol{\beta}^*\|_2^2 + \frac{1}{2\eta}\mathrm{tr}(\mathrm{G}_T^{1/2})\|\boldsymbol{\beta}^*\|_2^2 + \eta\mathrm{tr}(\mathrm{G}_T^{1/2}).$$

ADA-FULL incorporated into the composite mirror descent method ensures

$$R(T) \le \frac{\sigma}{2\eta}\|\boldsymbol{\beta}_*\|_2^2 + \frac{1}{2\eta}\max_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2\mathrm{tr}(\mathrm{G}_T^{1/2}) + \eta\mathrm{tr}(\mathrm{G}_T^{1/2}).$$

Compared with the Theorem 3, we confirm that our methods enjoy the regret bound close to that of ADA-FULL, and we point out the slight differences as following. First, the regret bounds of our two methods contain an additional term, respectively $\frac{\sum_{t=1}^T\sqrt{\sigma_t}}{2\eta}\max_{t\le T}\|\boldsymbol{\beta}_{t+1}\|_2^2$ and $\frac{\tau\sum_{t=1}^T\sqrt{\sigma_t}}{2\eta}\max_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2$. According to Cauchy-Schwarz inequality and $\Delta_T = \sum_{i=1}^T \sigma_i$, we have

$$\sum_{t=1}^T\sqrt{\sigma_t} \le \sqrt{T}\sqrt{\sum_{t=1}^T\sigma_t} = \sqrt{T\Delta_T},$$

which means the additional term does not affect the order of the regret bounds when Δ_T is small. Second, our two bounds magnify the third term by $\max\left(1, \frac{\max_{t\le T}\|\mathbf{g}_t\|_2 + \sqrt{\Delta_T}}{\delta}\right)$ and $\max\left(1, \frac{\sqrt{\Delta_T}}{\delta}\right)$ respectively, which does not change the order of the regret bounds when Δ_T is small. Let C_t^k denote the minimizer of $\|\mathrm{C}_t - \mathrm{C}_t^k\|_F$ over all rank k matrices. According to the property of frequent directions, we have

$$\Delta_T \le \|\mathrm{C}_T - \mathrm{C}_T^k\|_F^2/(\tau - k),$$

for any $k < \tau$, which means Δ_T is small when the outer product matrix of gradients is approximately low-rank. Moreover, when G_T is low-rank and $\mathrm{rank}(\mathrm{G}_T) = r$, we have $\Delta_T = 0$ by choosing $\tau = r + 1$.

Additionally, according to Proposition 2 of Krummenacher *et al.* [27], the regret bound of ADA-LR is also close to that of ADA-FULL. However, it only holds with high probability and requires $\tau \ge O(\log d)$. In contrast, the regret bounds of our methods are deterministic and hold for any integer $\tau > 0$.

Algorithm 2. Adaptive Mirror Descent via FD

```
1: Input: $\eta > 0, \delta > 0, \tau, \mathrm{S}_0 = \mathbf{0}_{\tau\times d}, \boldsymbol{\beta}_1 = \mathbf{0}$
2: for $t = 1, \ldots, T$ do
3:     Receive the gradient $\mathbf{g}_t = \nabla f_t(\boldsymbol{\beta}_t)$
4:     Insert $\mathbf{g}_t$ into the last row of $\mathrm{S}_{t-1}$
5:     Compute $\mathrm{U\Sigma V}^\top = \mathrm{S}_{t-1}$, $\sigma_t = \Sigma^2_{\tau\tau}$
6:     Compute $\mathrm{S}_t = \Sigma'\mathrm{V}^\top$ where $\Sigma' = \sqrt{\Sigma^2 - \sigma_t \mathrm{I}_\tau}$
7:     $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \frac{\eta}{\delta}\left(\mathbf{g}_t - \mathrm{V}(\delta\mathrm{I}_\tau + \Sigma')^{-1}\Sigma'\mathrm{V}^\top\mathbf{g}_t\right)$
8: end for
```

3.2 Adaptive Online Learning via FFD

Although the time complexity of ADA-FD is linear in the dimensionality d, it still suffers the quadratic dependence on the sketching size τ. Motivated by the fast implementation of frequent directions (5), we further accelerate our ADA-FD by doubling the sketching size and reducing the number of time-consuming computations. However, fast frequent directions cannot be directly applied to our methods, because it only computes the SVD of the sketch matrix S_t once every $\tau+1$ iterations while for updating the decision, our methods need to compute the square root of $\mathrm{S}_t^\top \mathrm{S}_t$ and the inverse of H_t by the SVD of S_t in each iteration. As a result, besides the doubling trick used in fast frequent directions, we carefully design a more efficient strategy to compute the SVD of S_t as explained below, which is motivated by the idea of incremental SVD [43].

Algorithm 3. Adaptive Dual Averaging via FFD

```
Input: $\eta > 0, \delta > 0, \tau, r_0 = 0, \mathrm{V}_0 = \mathbf{0}_{d\times 2\tau}, \mathrm{M}_0 = \mathbf{0}_{2\tau\times 2\tau}, \bar{\mathbf{g}}_0 = \mathbf{0}, \boldsymbol{\beta}_1 = \mathbf{0}$
for $t = 1, \ldots, T$ do
  Receive the gradient $\mathbf{g}_t = \nabla f_t(\boldsymbol{\beta}_t)$
  Compute $\bar{\mathbf{g}}_t = \bar{\mathbf{g}}_{t-1} + \mathbf{g}_t$ and $\mathbf{g}' = \mathrm{V}_{t-1}(\mathrm{V}_{t-1}^\top \mathbf{g}_t)$
  if $\mathbf{g}' \neq \mathbf{g}_t$ then
    Set $r_{t-1} = r_{t-1} + 1$ and $\mathbf{v}_{r_{t-1}}^{t-1} = \frac{\mathbf{g}_t - \mathbf{g}'}{\|\mathbf{g}_t - \mathbf{g}'\|_2}$
  end if
  Set $r_t = r_{t-1}, \mathrm{V}_t = \mathrm{V}_{t-1}, \sigma_t = 0$
  Compute $\mathrm{M}_t = \mathrm{M}_{t-1} + (\mathrm{V}_{t-1}^\top \mathbf{g}_t)(\mathrm{V}_{t-1}^\top \mathbf{g}_t)^\top$
  Compute $\mathrm{U}\Sigma\mathrm{U}^\top = \mathrm{M}_t$
  $\boldsymbol{\beta}_{t+1} = -\frac{\eta}{\delta}\left(\bar{\mathbf{g}}_t - \mathrm{V}_t\mathrm{U}(\delta\mathrm{I}_{2\tau} + \Sigma^{1/2})^{-1}\Sigma^{1/2}\mathrm{U}^\top\mathrm{V}_t^\top\bar{\mathbf{g}}_t\right)$
  if $r_t = 2\tau$ then
    Set $\sigma_t = \Sigma_{\tau\tau}, \mathrm{M}_t = \max(\Sigma - \sigma_t\mathrm{I}_{2\tau}, 0), \mathrm{V}_t = \mathrm{V}_t\mathrm{U}$
    Set $r_t = \tau - 1$ and $\{\mathbf{v}_i^t | i = r_t + 1, \ldots, 2\tau\}$ to be zero
  end if
end for
```

According to the doubling trick, the size of S_t is changed from $\tau \times d$ to $2\tau \times d$. Moreover, instead of directly maintaining S_t, we maintain two matrices M_t and V_t which satisfy

$$\mathrm{M}_t^{1/2}\mathrm{V}_t^\top = \mathrm{S}_t \in \mathbb{R}^{2\tau\times d},$$

where $\mathrm{M}_t \in \mathbb{R}^{2\tau\times 2\tau}$ is a symmetric matrix and $\mathrm{V}_t = [\mathbf{v}_1^t, \mathbf{v}_2^t, \ldots, \mathbf{v}_{2\tau}^t] \in \mathbb{R}^{d\times 2\tau}$ consists of $r_t \leq 2\tau$ orthonormal vectors and $2\tau - r_t$ zero vectors. For $t = 0$, we simply set

$$r_0 = 0, \mathrm{V}_0 = \mathbf{0}_{d\times 2\tau}, \mathrm{M}_0 = \mathbf{0}_{2\tau\times 2\tau}.$$

In each iteration $t = 1, \ldots, T$, after receiving the gradient $\mathbf{g}_t$, we first compute

$$\mathbf{g}' = \mathrm{V}_{t-1}(\mathrm{V}_{t-1}^\top \mathbf{g}_t).$$

If $\mathbf{g}' \neq \mathbf{g}_t$, we set $r_{t-1} = r_{t-1} + 1$ and

$$\mathbf{v}_{r_{t-1}}^{t-1} = \frac{\mathbf{g}_t - \mathbf{g}'}{\|\mathbf{g}_t - \mathbf{g}'\|_2},$$

where $(\mathbf{v}_{r_{t-1}}^{t-1})^\top \mathbf{v}_i^{t-1} = 0$ for any $1 \leq i < r_{t-1}$. Then, we have $\mathrm{V}_{t-1}\mathrm{V}_{t-1}^\top \mathbf{g}_t = \mathbf{g}_t$, which leads to

$$\begin{aligned}
&\mathrm{V}_{t-1}\mathrm{M}_{t-1}\mathrm{V}_{t-1}^\top + \mathbf{g}_t\mathbf{g}_t^\top \\
&\quad = \mathrm{V}_{t-1}\mathrm{M}_{t-1}\mathrm{V}_{t-1}^\top + \mathrm{V}_{t-1}\mathrm{V}_{t-1}^\top\mathbf{g}_t\mathbf{g}_t^\top\mathrm{V}_{t-1}\mathrm{V}_{t-1}^\top \\
&\quad = \mathrm{V}_{t-1}(\mathrm{M}_{t-1} + \mathrm{V}_{t-1}^\top\mathbf{g}_t\mathbf{g}_t^\top\mathrm{V}_{t-1})\mathrm{V}_{t-1}^\top.
\end{aligned}$$

So, we can set $r_t = r_{t-1}, \mathrm{V}_t = \mathrm{V}_{t-1}$ and compute M_t as

$$\mathrm{M}_t = \mathrm{M}_{t-1} + (\mathrm{V}_{t-1}^\top\mathbf{g}_t)(\mathrm{V}_{t-1}^\top\mathbf{g}_t)^\top.$$

Now, we can directly utilize the above two matrices V_t and M_t to update the decision. Specifically, we first efficiently compute the SVD of M_t as

$$\mathrm{U}\Sigma\mathrm{U}^\top = \mathrm{M}_t.$$

Then, we have

$$\begin{aligned}
(\mathrm{S}_t\mathrm{S}_t^\top)^{1/2} &= (\mathrm{V}_t\mathrm{M}_t\mathrm{V}_t^\top)^{1/2} \\
&= (\mathrm{V}_t\mathrm{U}\Sigma\mathrm{U}^\top\mathrm{V}_t^\top)^{1/2} \\
&= \mathrm{V}_t\mathrm{U}\Sigma^{1/2}\mathrm{U}^\top\mathrm{V}_t^\top,
\end{aligned}$$

which means that β_{t+1} can be computed efficiently with Woodbury formula as in ADA-FD. Moreover, according to fast frequent directions, if $r_t = 2\tau$, we need to make the last $\tau + 1$ rows of S_t equal to zero by computing

$$\mathrm{M}_t = \max(\Sigma - \sigma_t\mathrm{I}_{2\tau}, 0), \mathrm{V}_t = \mathrm{V}_t\mathrm{U},$$

where $\sigma_t = \Sigma_{\tau\tau}$, and setting $r_t = \tau - 1$ and the right $\tau + 1$ columns of V_t to be zero. In this way, the rank of $\mathrm{M}_t^{1/2}\mathrm{V}_t^\top$ is reduced to $\tau - 1$, and would grow to 2τ in subsequent iterations. So, there could exist *cyclical changes* about it.

With the above procedures, we propose our second efficient variant of ADA-FULL, named as adaptive online learning via fast frequent directions (ADA-FFD). Furthermore, the detailed procedures of ADA-FFD for the primal-dual subgradient method and composite mirror descent method are summarized in Algorithms 3 and 4, respectively.

Algorithm 4. Adaptive Mirror Descent via FFD

```
Input: $\eta > 0, \delta > 0, \tau, r_0 = 0, \mathrm{V}_0 = \mathbf{0}_{d\times 2\tau}, \mathrm{M}_0 = \mathbf{0}_{2\tau\times 2\tau}, \boldsymbol{\beta}_1 = \mathbf{0}$
for $t = 1, \ldots, T$ do
  Receive the gradient $\mathbf{g}_t = \nabla f_t(\boldsymbol{\beta}_t)$
  Compute $\mathbf{g}' = \mathrm{V}_{t-1}(\mathrm{V}_{t-1}^\top \mathbf{g}_t)$
  if $\mathbf{g}' \neq \mathbf{g}_t$ then
    Set $r_{t-1} = r_{t-1} + 1$ and $\mathbf{v}_{r_{t-1}}^{t-1} = \frac{\mathbf{g}_t - \mathbf{g}'}{\|\mathbf{g}_t - \mathbf{g}'\|_2}$
  end if
  Set $r_t = r_{t-1}, \mathrm{V}_t = \mathrm{V}_{t-1}, \sigma_t = 0$
  Compute $\mathrm{M}_t = \mathrm{M}_{t-1} + (\mathrm{V}_{t-1}^\top \mathbf{g}_t)(\mathrm{V}_{t-1}^\top \mathbf{g}_t)^\top$
  Compute $\mathrm{U}\Sigma\mathrm{U}^\top = \mathrm{M}_t$
  $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \frac{\eta}{\delta}\left(\mathbf{g}_t - \mathrm{V}_t\mathrm{U}(\delta\mathrm{I}_{2\tau} + \Sigma^{1/2})^{-1}\Sigma^{1/2}\mathrm{U}^\top\mathrm{V}_t^\top\mathbf{g}_t\right)$
  if $r_t = 2\tau$ then
    Set $\sigma_t = \Sigma_{\tau\tau}, \mathrm{M}_t = \max(\Sigma - \sigma_t\mathrm{I}_{2\tau}, 0), \mathrm{V}_t = \mathrm{V}_t\mathrm{U}$
    Set $r_t = \tau - 1$ and $\{\mathbf{v}_i^t | i = r_t + 1, \ldots, 2\tau\}$ to be zero
  end if
end for
```

3.2.1 Computational Complexity

In each iteration, our Algorithms 3 and 4 need to compute $\mathbf{g}'$, $\mathbf{v}_{r_{t-1}}^{t-1}$, M_t, the SVD of M_t, and $\boldsymbol{\beta}_{t+1}$, the time complexities of which are $O(\tau d)$, $O(d)$, $O(\tau d)$, $O(\tau^3)$, and $O(\tau d)$, respectively. Assuming $\tau \leq \sqrt{d}$, in each iteration, the total time complexity of these computations is $O(\tau d)$. Besides these computations, Algorithms 3 and 4 need to compute $\mathrm{V_t} = \mathrm{V_t U}$ once every $\tau + 1$ iterations, which causes the time complexity of $O(\tau^2 d)$. Therefore, in each iteration, the average time complexities of Algorithms 3 and 4 are $O\left(\frac{\tau dT+\tau^2 d(T/\tau)}{T}\right) = O(\tau d)$ which is linear in both the dimensionality d and the sketching size τ. Meanwhile, the space complexities of Algorithms 3 and 4 are still $O(\tau d)$ caused by the maintenance of V_t, because ADA-FFD only doubles the sketching size of ADA-FD. Compared with ADA-FD and RADAGRAD, our ADA-FFD further reduces the time complexity to $O(\tau d)$ while enjoys the same space complexity of $O(\tau d)$.

3.2.2 Theoretical Guarantees

Note that Algorithms 3 and 4 enjoy almost the same regret bounds as Algorithms 1 and 2, respectively. For completeness, we present the regret bounds of Algorithms 3 and 4 in the following two theorems, respectively.

Theorem 4. *Adaptive dual averaging via fast frequent directions ensures*

$$\begin{aligned} R(T) \leq & \frac{\delta}{2\eta}\|\boldsymbol{\beta}^*\|_2^2 + \frac{1}{2\eta}\mathrm{tr}(\mathrm{G}_T^{1/2})\|\boldsymbol{\beta}^*\|_2^2 \\ & + \eta\mathrm{max}\left(1, \frac{\mathrm{max}_{t\leq T}\|\mathbf{g}_t\|_2 + \sqrt{\Delta_T}}{\delta}\right)\mathrm{tr}(\mathrm{G}_T^{1/2}) \\ & + \frac{\sum_{t=1}^T \sqrt{\sigma_t}}{2\eta}\mathrm{max}_{t\leq T}\|\boldsymbol{\beta}_{t+1}\|_2^2, \end{aligned} \tag{11}$$

where $\Delta_T = \sum_{i=1}^T \sigma_i$.

Theorem 5. *Adaptive mirror descent via fast frequent directions ensures*

$$\begin{aligned} R(T) \leq & \frac{\delta}{2\eta}\|\boldsymbol{\beta}_*\|_2^2 + \frac{1}{2\eta}\mathrm{max}_{t\leq T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2\mathrm{tr}(\mathrm{G}_T^{1/2}) \\ & + \eta\mathrm{max}\left(1, \frac{\sqrt{\Delta_T}}{\delta}\right)\mathrm{tr}(\mathrm{G}_T^{1/2}) \\ & + \frac{\tau\sum_{t=1}^T \sqrt{\sigma_t}}{\eta}\mathrm{max}_{t\leq T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2, \end{aligned} \tag{12}$$

where $\Delta_T = \sum_{i=1}^T \sigma_i$.

According to the property of fast frequent directions proved by Ghashami *et al.* [31], we still have

$$\Delta_T \leq \|\mathrm{C}_T - \mathrm{C}_T^k\|_F^2/(\tau - k), \tag{13}$$

for any $k < \tau$. At first sight, (11) is the same as (9) in Theorem 1 and (12) only doubles the last term of (10) in Theorem 2. However, the sketching size of ADA-FFD is 2τ. For a fair comparison, we should reduce it to τ, which implies that τ in (12) and (13) would be replaced with $\tau/2$.

3.3 Discussions

We would like to emphasize the differences between our work and Krummenacher *et al.* [27]. First, both ADA-LR and RADAGRAD only consider the composite mirror descent method, ignoring the primal-dual subgradient method. Second, unlike random projections adopted in Krummenacher *et al.* [27], our work makes use of frequent directions and fast frequent directions that belong to deterministic matrix sketching techniques. As a result, the theoretical guarantees of our methods are deterministic which are more robust than the probabilistic regret bound of ADA-LR. Third, our methods are very efficient in the sense that the time and storage complexities are linear in the dimensionality d, without losing the regret bound. Especially, the time complexities of our Algorithms 3 and 4 are also linear in the sketching size τ. By contrast, RADAGRAD only reaches similar complexities as our Algorithms 1 and 2 while does not have theoretical justifications.

Next, we discuss the differences between our work and Luo *et al.* [30]. First, our methods and Luo *et al.* [30] are designed for different tasks. Our goal is to develop an efficient variant of ADA-FULL, and the goal of Luo *et al.* [30] is to accelerate ONS. Note that ADA-FULL is a data dependent algorithm for general convex functions and ONS is proposed to derive a logarithmic regret bound for exponentially concave functions. Second, the theoretical analysis in our work is obviously different from Luo *et al.* [30]. They provide a regret bound of ONS with order $O(\sqrt{Td})$ for general convex functions by setting $\eta_t = O(1/\sqrt{t})$. Although its update rules (6) can be reformulated as

$$\mathrm{H}_t = \delta\mathrm{I}_d + \sum_{i=1}^t \frac{1}{\sqrt{i}}\mathbf{g}_i\mathbf{g}_i^\top \text{ and } \boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta\mathrm{H}_t^{-1}\mathbf{g}_t,$$

which is similar to ADA-FULL, its $O(\sqrt{Td})$ bound destroys the data dependent property. Third, the main algorithm of Luo *et al.* [30] can be regarded as a mirror descent method. In contrast, our work proposes methods for both the primal-dual subgradient method and composite mirror descent method, respectively.

4 Theoretical Analysis

In this section, we provide the proofs of Theorems 1 and 2 and omit the proofs of Theorems 4 and 5, because it is not difficult to prove Theorems 4 and 5 in the same way.

4.1 Supporting Results

The following results are used throughout our analysis.

Lemma 1 (Variant of Proposition 2 in Duchi *et al.* [26]). *Let the sequence* $\{\boldsymbol{\beta}_t\}$ *be generated by Algorithm 1. We have*

$$\begin{aligned} R(T) \leq & \frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*) + \frac{\eta}{2}\sum_{t=1}^T \|f_t'(\boldsymbol{\beta}_t)\|_{\Psi_{t-1}^*}^2 \\ & + \frac{\sum_{t=1}^T \sqrt{\sigma_t}}{2\eta}\mathrm{max}_{t\leq T}\|\boldsymbol{\beta}_{t+1}\|_2^2. \end{aligned}$$

Lemma 1 can be regarded as a variant of Proposition 2 in Duchi *et al.* [26], when the condition $\Psi_{t+1}(\boldsymbol{\beta}) \geq \Psi_t(\boldsymbol{\beta})$ cannot be met due to $\mathrm{H}_{t+1}\nsucceq\mathrm{H}_t$ in this work. And it can be derived from the proof of Proposition 2 in Duchi *et al.* [26] with slight modifications to deal with $\Psi_{t+1}(\boldsymbol{\beta})\ngeq\Psi_t(\boldsymbol{\beta})$. We include the proof for completeness.

Proof. The conjugate dual of $t\varphi(\boldsymbol{\beta}) + \frac{1}{\eta}\Psi_t(\boldsymbol{\beta})$ is defined by

$$\Phi_t^*(\mathbf{g}) = \sup\nolimits_{\boldsymbol{\beta}}\left\{\langle \mathbf{g}, \boldsymbol{\beta}\rangle - t\varphi(\boldsymbol{\beta}) - \frac{1}{\eta}\Psi_t(\boldsymbol{\beta})\right\}.$$

Thus, the gradient of $\Phi_t^*(\mathbf{g})$ can be calculated as

$$\nabla\Phi_t^*(\mathbf{g}) = \arg\min_{\boldsymbol{\beta}}\left\{-\langle \mathbf{g}, \boldsymbol{\beta}\rangle + t\varphi(\boldsymbol{\beta}) + \frac{1}{\eta}\Psi_t(\boldsymbol{\beta})\right\}. \tag{14}$$

Because $\frac{1}{\eta}\Psi_t(\boldsymbol{\beta})$ is $\frac{1}{\eta}$-strongly convex with respect to the norm $\|\cdot\|_{\Psi_t}$, we have $\|\nabla\Phi_t^*(\mathbf{x}) - \nabla\Phi_t^*(\mathbf{y})\|_{\Psi_t} \le \eta\|\mathbf{x}-\mathbf{y}\|_{\Psi_t^*}$ which means the function Φ_t^* has η-Lipschitz continuous gradients with respect to $\|\cdot\|_{\Psi_t^*}$. Further, we have

$$\Phi_t^*(\mathbf{y}) \le \Phi_t^*(\mathbf{x}) + \big\langle \nabla\Phi_t^*(\mathbf{x}), \mathbf{y}-\mathbf{x}\big\rangle + \frac{\eta}{2}\|\mathbf{y}-\mathbf{x}\|_{\Psi_t^*}^2. \tag{15}$$

Both the identity (14) and the bound (15) were used in the proof of Proposition 2 in Duchi *et al.* [26]. In order to complete the proof, we introduce an inequality

$$\begin{aligned}&\sum_{t=1}^{T}\{f_t(\boldsymbol{\beta}_t) + \varphi(\boldsymbol{\beta}_t) - f_t(\boldsymbol{\beta}^*) - \varphi(\boldsymbol{\beta}^*)\}\\ &\quad\le \frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*) + \sum_{t=1}^{T}\big\{\big\langle \mathbf{g}_t, \boldsymbol{\beta}_t\big\rangle + \varphi(\boldsymbol{\beta}_t)\big\} + \Phi_T^*(-\bar{\mathbf{g}}_T),\end{aligned} \tag{16}$$

from the proof of Proposition 2 in Duchi *et al.* [26] again.

Let $\Sigma_t, \Sigma_t', \mathrm{V}_t$ denote the values of $\Sigma, \Sigma', \mathrm{V}$ in the tth round of Algorithm 1. For each $t = 1, \ldots, T$, we have

$$\begin{aligned}&(\mathrm{S}_t^\top\mathrm{S}_t)^{1/2} - (\mathrm{S}_{t-1}^\top\mathrm{S}_{t-1})^{1/2} + \sqrt{\sigma_t}V_tV_t^\top\\ &\quad= \mathrm{V}_t\Sigma_t'\mathrm{V}_t^\top + \sqrt{\sigma_t}V_tV_t^\top - (\mathrm{S}_{t-1}^\top\mathrm{S}_{t-1})^{1/2}\\ &\quad\succeq \mathrm{V}_t\Sigma_t\mathrm{V}_t^\top - (\mathrm{S}_{t-1}^\top\mathrm{S}_{t-1})^{1/2}\\ &\quad= (\mathrm{S}_{t-1}^\top\mathrm{S}_{t-1} + \mathbf{g}_t\mathbf{g}_t^\top)^{1/2} - (\mathrm{S}_{t-1}^\top\mathrm{S}_{t-1})^{1/2}\\ &\quad\succeq 0,\end{aligned}$$

which implies $-\Psi_t(\mathbf{x}) \le -\Psi_{t-1}(\mathbf{x}) + \frac{\sqrt{\sigma_t}}{2}\|\mathbf{x}\|_2^2$.

Thus, we have

$$\begin{aligned}\Phi_T^*(-\bar{\mathbf{g}}_T) &= -\big\langle \bar{\mathbf{g}}_T, \boldsymbol{\beta}_{T+1}\big\rangle - T\varphi(\boldsymbol{\beta}_{T+1}) - \frac{1}{\eta}\Psi_T(\boldsymbol{\beta}_{T+1})\\ &\le -\big\langle \bar{\mathbf{g}}_T, \boldsymbol{\beta}_{T+1}\big\rangle - T\varphi(\boldsymbol{\beta}_{T+1}) - \frac{1}{\eta}\Psi_{T-1}(\boldsymbol{\beta}_{T+1})\\ &\quad+ \frac{\sqrt{\sigma_T}}{2\eta}\|\boldsymbol{\beta}_{T+1}\|_2^2\\ &\le \sup\nolimits_{\boldsymbol{\beta}}\left(-\big\langle \bar{\mathbf{g}}_T, \boldsymbol{\beta}\big\rangle - (T-1)\varphi(\boldsymbol{\beta}) - \frac{1}{\eta}\Psi_{T-1}(\boldsymbol{\beta})\right)\\ &\quad- \varphi(\boldsymbol{\beta}_{T+1}) + \frac{\sqrt{\sigma_T}}{2\eta}\|\boldsymbol{\beta}_{T+1}\|_2^2\\ &= \Phi_{T-1}^*(-\bar{\mathbf{g}}_T) - \varphi(\boldsymbol{\beta}_{T+1}) + \frac{\sqrt{\sigma_T}}{2\eta}\|\boldsymbol{\beta}_{T+1}\|_2^2,\end{aligned}$$

which contains an additional term $\frac{\sqrt{\sigma_T}}{2\eta}\|\boldsymbol{\beta}_{T+1}\|_2^2$ caused by $\mathrm{H}_T \not\succeq \mathrm{H}_{T-1}$ comparing with Duchi *et al.* [26].

Using the identity (14), the bound (15) and the inequality (16), we have

$$\begin{aligned}&\sum_{t=1}^{T}\{f_t(\boldsymbol{\beta}_t) + \varphi(\boldsymbol{\beta}_{t+1}) - f_t(\boldsymbol{\beta}^*) - \varphi(\boldsymbol{\beta}^*)\}\\ &\quad\le \frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*) + \sum_{t=1}^{T}\big\{\big\langle \mathbf{g}_t, \boldsymbol{\beta}_t\big\rangle + \varphi(\boldsymbol{\beta}_{t+1})\big\}\\ &\qquad+ \Phi_{T-1}^*(-\bar{\mathbf{g}}_T) - \varphi(\boldsymbol{\beta}_{T+1}) + \frac{\sqrt{\sigma_T}}{2\eta}\|\boldsymbol{\beta}_{T+1}\|_2^2\\ &\quad\le \frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*) + \sum_{t=1}^{T}\big\{\big\langle \mathbf{g}_t, \boldsymbol{\beta}_t\big\rangle + \varphi(\boldsymbol{\beta}_{t+1})\big\}\\ &\qquad+ \Phi_{T-1}^*(-\bar{\mathbf{g}}_{T-1}) - \big\langle \nabla\Phi_{T-1}^*(-\bar{\mathbf{g}}_{T-1}), \mathbf{g}_T\big\rangle\\ &\qquad+ \frac{\eta}{2}\|\mathbf{g}_T\|_{\Psi_{T-1}^*}^2 - \varphi(\boldsymbol{\beta}_{T+1}) + \frac{\sqrt{\sigma_T}}{2\eta}\|\boldsymbol{\beta}_{T+1}\|_2^2\\ &\quad= \frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*) + \sum_{t=1}^{T-1}\big\{\big\langle \mathbf{g}_t, \boldsymbol{\beta}_t\big\rangle + \varphi(\boldsymbol{\beta}_{t+1})\big\}\\ &\qquad+ \Phi_{T-1}^*(-\bar{\mathbf{g}}_{T-1}) + \frac{\eta}{2}\|\mathbf{g}_T\|_{\Psi_{T-1}^*}^2 + \frac{\sqrt{\sigma_T}}{2\eta}\|\boldsymbol{\beta}_{T+1}\|_2^2.\end{aligned}$$

By repeating the above steps, we have

$$\begin{aligned}&\sum_{t=1}^{T}\{f_t(\boldsymbol{\beta}_t) + \varphi(\boldsymbol{\beta}_{t+1}) - f_t(\boldsymbol{\beta}^*) - \varphi(\boldsymbol{\beta}^*)\}\\ &\quad\le \frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*) + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{g}_t\|_{\Psi_{t-1}^*}^2 + \sum_{t=1}^{T}\frac{\sqrt{\sigma_t}}{2\eta}\|\boldsymbol{\beta}_{t+1}\|_2^2\\ &\qquad+ \Phi_0^*(-\bar{\mathbf{g}}_0).\end{aligned}$$

Note that $\varphi(\boldsymbol{\beta}) = 0$ and $\Phi_0^*(\mathbf{0}) = 0$. We complete the proof. □

Lemma 2 (Proposition 3 in Duchi *et al.* [26]). *Let the sequence* $\{\boldsymbol{\beta}_t\}$ *be generated by Algorithm 2. We have*

$$\begin{aligned}R(T) \le &\frac{1}{\eta}\sum_{t=1}^{T-1}[B_{\Psi_{t+1}}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_{t+1}) - B_{\Psi_t}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_{t+1})]\\ &+ \frac{1}{\eta}B_{\Psi_1}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_1) + \frac{\eta}{2}\sum_{t=1}^{T}\|f_t'(\boldsymbol{\beta}_t)\|_{\Psi_t^*}^2.\end{aligned}$$

Lemma 3 (Lemma 10 in Duchi *et al.* [26]). *Let* $\mathrm{G}_t = \sum_{i=1}^{t}\mathbf{g}_i\mathbf{g}_i^\top$ *and* $\mathrm{A}^\dagger$ *denote the pseudo-inverse of* A*, then*

$$\sum_{t=1}^{T}\langle \mathbf{g}_t, (\mathrm{G}_t^{1/2})^\dagger\mathbf{g}_t\rangle \le 2\sum_{t=1}^{T}\langle \mathbf{g}_t, (\mathrm{G}_T^{1/2})^\dagger\mathbf{g}_t\rangle = 2\mathrm{tr}(\mathrm{G}_T^{1/2}).$$

Lemma 4 (Derived From Theorem 3.1 and Its Proof in Ghashami *et al.* [31]). *Let* $\Delta_t = \sum_{i=1}^{t}\sigma_i$*. In Algorithms 1 and 2,* S_t *is the sketch generated by performing frequent directions on the input* C_t*. Then for any* t *and* $k < \tau$, $\mathrm{C}_t^\top\mathrm{C}_t \succeq \mathrm{S}_t^\top\mathrm{S}_t \succeq \mathrm{C}_t^\top\mathrm{C}_t - \Delta_t\mathrm{I}_d$ *and* $\Delta_t \le \|\mathrm{C}_t - \mathrm{C}_t^k\|_F^2/(\tau - k)$*, where* C_t^k *denotes the minimizer of* $\|\mathrm{C}_t - \mathrm{C}_t^k\|_F$ *over all rank* k *matrices.*

4.2 Proof of Theorem 1

We first consider $\frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*)$ in the upper bound of Lemma 1. We have

$$
\begin{aligned}
\frac{1}{\eta}\Psi_T(\boldsymbol{\beta}^*) &= \frac{1}{2\eta}\left\langle \boldsymbol{\beta}^*, (\delta \mathrm{I}_d + (\mathrm{S}_T^\top \mathrm{S}_T)^{1/2})\boldsymbol{\beta}^*\right\rangle \\
&\le \frac{\delta}{2\eta}\|\boldsymbol{\beta}^*\|_2^2 + \frac{1}{2\eta}\left\langle \boldsymbol{\beta}^*, (\mathrm{C}_T^\top \mathrm{C}_T)^{1/2}\boldsymbol{\beta}^*\right\rangle \\
&\le \frac{\delta}{2\eta}\|\boldsymbol{\beta}^*\|_2^2 + \frac{1}{2\eta}\lambda_{\max}(\mathrm{G}_T^{1/2})\|\boldsymbol{\beta}^*\|_2^2 \\
&\le \frac{\delta}{2\eta}\|\boldsymbol{\beta}^*\|_2^2 + \frac{1}{2\eta}\mathrm{tr}(\mathrm{G}_T^{1/2})\|\boldsymbol{\beta}^*\|_2^2,
\end{aligned}
\tag{17}
$$

where $\lambda_{\max}(\mathrm{G}_T^{1/2})$ denotes the largest eigenvalue of $\mathrm{G}_T^{1/2}$.

Before considering $\frac{\eta}{2}\sum_{t=1}^T \|f_t'(\boldsymbol{\beta}_t)\|^2_{\Psi^*_{t-1}}$, we need to derive the lower bound of H_{t-1}. Let $c = \frac{\delta}{\|\mathbf{g}_t\|_2 + \sqrt{\Delta_{t-1}}}$. If $c < 1$, we have

$$
\begin{aligned}
\mathrm{H}_{t-1} &= \delta \mathrm{I}_d + (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} \\
&\succeq c(\|\mathbf{g}_t\|_2 \mathrm{I}_d + \sqrt{\Delta_{t-1}}\mathrm{I}_d + (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2}) \\
&\succeq c(\|\mathbf{g}_t\|_2 \mathrm{I}_d + (\Delta_{t-1}\mathrm{I}_d + \mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2}) \\
&\succeq c(\|\mathbf{g}_t\|_2 \mathrm{I}_d + (\mathrm{C}_{t-1}^\top \mathrm{C}_{t-1})^{1/2}) \\
&\succeq c(\mathrm{C}_{t-1}^\top \mathrm{C}_{t-1} + \|\mathbf{g}_t\|_2^2 \mathrm{I}_d)^{1/2} \\
&\succeq c(\mathrm{C}_t^\top \mathrm{C}_t)^{1/2},
\end{aligned}
$$

where the second inequality is due to $\sqrt{\Delta_{t-1}} + x \ge \sqrt{\Delta_{t-1} + x^2}$ for any $x \ge 0$ and the third inequality is due to Lemma 4.

In the other case $\delta \ge \sqrt{\Delta_{t-1}} + \|\mathbf{g}_t\|_2$, we have

$$
\begin{aligned}
\mathrm{H}_{t-1} &= \delta \mathrm{I}_d + (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} \\
&\succeq \|\mathbf{g}_t\|_2 \mathrm{I}_d + \sqrt{\Delta_{t-1}}\mathrm{I}_d + (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} \\
&\succeq \|\mathbf{g}_t\|_2 \mathrm{I}_d + (\Delta_{t-1}\mathrm{I}_d + \mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} \\
&\succeq \|\mathbf{g}_t\|_2 \mathrm{I}_d + (\mathrm{C}_{t-1}^\top \mathrm{C}_{t-1})^{1/2} \\
&\succeq (\mathrm{C}_t^\top \mathrm{C}_t)^{1/2}.
\end{aligned}
$$

Thus, for any $\delta > 0$, we have

$$
\mathrm{H}_{t-1} \succeq \min\left(1, \frac{\delta}{\|\mathbf{g}_t\|_2 + \sqrt{\Delta_{t-1}}}\right)(\mathrm{C}_t^\top \mathrm{C}_t)^{1/2}.
$$

Then, we have

$$
\begin{aligned}
&\sum_{t=1}^T \|f_t'(\boldsymbol{\beta}_t)\|^2_{\Psi^*_{t-1}} \\
&= \sum_{t=1}^T \left\langle \mathbf{g}_t, (\mathrm{H}_{t-1})^{-1}\mathbf{g}_t \right\rangle \\
&\le \sum_{t=1}^T \max\left(1, \frac{\|\mathbf{g}_t\|_2 + \sqrt{\Delta_{t-1}}}{\delta}\right)\left\langle \mathbf{g}_t, (\mathrm{G}_t^\dagger)^{1/2}\mathbf{g}_t \right\rangle \\
&\le \max\left(1, \frac{\max_{t\le T}\|\mathbf{g}_t\|_2 + \sqrt{\Delta_T}}{\delta}\right)\sum_{t=1}^T \left\langle \mathbf{g}_t, (\mathrm{G}_t^\dagger)^{1/2}\mathbf{g}_t \right\rangle \\
&\le 2\max\left(1, \frac{\max_{t\le T}\|\mathbf{g}_t\|_2 + \sqrt{\Delta_T}}{\delta}\right)\mathrm{tr}(\mathrm{G}_T^{1/2}),
\end{aligned}
\tag{18}
$$

where the last inequality is due to Lemma 3.

We complete the proof by substituting (17) and (18) into Lemma 1.

4.3 Proof of Theorem 2

Let $\Sigma_t, \Sigma_t', \mathrm{V}_t$ denote the values of $\Sigma, \Sigma', \mathrm{V}$ in the tth round of Algorithm 2.

For each $t = 1, \ldots, T$, we have

$$
\begin{aligned}
&(\mathrm{S}_t^\top \mathrm{S}_t)^{1/2} - (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} + \sqrt{\sigma_t}V_tV_t^\top \\
&\quad = \mathrm{V}_t\Sigma_t'\mathrm{V}_t^\top + \sqrt{\sigma_t}V_tV_t^\top - (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} \\
&\quad \succeq \mathrm{V}_t\Sigma_t\mathrm{V}_t^\top - (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} \\
&\quad = (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1} + \mathbf{g}_t\mathbf{g}_t^\top)^{1/2} - (\mathrm{S}_{t-1}^\top \mathrm{S}_{t-1})^{1/2} \\
&\quad \succeq 0.
\end{aligned}
$$

Let $\widetilde{\mathrm{G}}_t = \mathrm{S}_t^\top \mathrm{S}_t$. Then, considering the first term in the upper bound of Lemma 2, we have

$$
\begin{aligned}
&B_{\Psi_{t+1}}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_{t+1}) - B_{\Psi_t}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_{t+1}) \\
&\quad = \frac{1}{2}\langle \boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1}, (\mathrm{H}_{t+1} - \mathrm{H}_t)(\boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1})\rangle \\
&\quad = \frac{1}{2}\left\langle \boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1}, (\widetilde{\mathrm{G}}_{t+1}^{1/2} - \widetilde{\mathrm{G}}_t^{1/2})(\boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1})\right\rangle \\
&\quad \le \frac{1}{2}\left\langle \boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1}, (\widetilde{\mathrm{G}}_{t+1}^{1/2} - \widetilde{\mathrm{G}}_t^{1/2})(\boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1})\right\rangle \\
&\qquad + \frac{1}{2}\langle \boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1}, \sqrt{\sigma_{t+1}}V_{t+1}V_{t+1}^\top(\boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1})\rangle \\
&\quad \le \frac{1}{2}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1}\|_2^2\lambda_{\max}(\widetilde{\mathrm{G}}_{t+1}^{1/2} - \widetilde{\mathrm{G}}_t^{1/2} + \sqrt{\sigma_{t+1}}V_{t+1}V_{t+1}^\top) \\
&\quad \le \frac{1}{2}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_{t+1}\|_2^2\mathrm{tr}(\widetilde{\mathrm{G}}_{t+1}^{1/2} - \widetilde{\mathrm{G}}_t^{1/2} + \sqrt{\sigma_{t+1}}V_{t+1}V_{t+1}^\top) \\
&\quad \le \frac{1}{2}\max\nolimits_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2\Big(\mathrm{tr}(\widetilde{\mathrm{G}}_{t+1}^{1/2} - \widetilde{\mathrm{G}}_t^{1/2}) + \tau\sqrt{\sigma_{t+1}}\Big).
\end{aligned}
$$

Note that $\boldsymbol{\beta}_1 = \mathbf{0}$. We further get

$$
\begin{aligned}
&\sum_{t=1}^{T-1}[B_{\Psi_{t+1}}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_{t+1}) - B_{\Psi_t}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_{t+1})] + B_{\Psi_1}(\boldsymbol{\beta}^*, \boldsymbol{\beta}_1) \\
&\quad \le \frac{1}{2}\max\nolimits_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2\mathrm{tr}(\widetilde{\mathrm{G}}_T^{1/2} - \widetilde{\mathrm{G}}_1^{1/2}) \\
&\qquad + \frac{\tau\sum_{t=2}^T\sqrt{\sigma_t}}{2}\max\nolimits_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2 + \frac{1}{2}\langle \boldsymbol{\beta}^*, \mathrm{H}_1\boldsymbol{\beta}^*\rangle \\
&\quad \le \frac{1}{2}\max\nolimits_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2\mathrm{tr}(\mathrm{G}_T^{1/2}) \\
&\qquad + \frac{\tau\sum_{t=1}^T\sqrt{\sigma_t}}{2}\max\nolimits_{t\le T}\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_t\|_2^2 + \frac{\delta}{2}\|\boldsymbol{\beta}^*\|_2^2,
\end{aligned}
\tag{19}
$$

where we use Lemma 4 in the last inequality.

Before considering $\sum_{t=1}^T \|f_t'(\boldsymbol{\beta}_t)\|^2_{\Psi^*_t}$, we need to derive the lower bound of H_t.

If $\delta < \sqrt{\Delta_t}$, we have

$$
\begin{aligned}
\mathrm{H}_t = \delta \mathrm{I}_d + (\mathrm{S}_t^\top \mathrm{S}_t)^{1/2} &\succeq \frac{\delta(\sqrt{\Delta_t}\mathrm{I}_d + (\mathrm{S}_t^\top \mathrm{S}_t)^{1/2})}{\sqrt{\Delta_t}} \\
&\succeq \frac{\delta(\Delta_t\mathrm{I}_d + \mathrm{S}_t^\top \mathrm{S}_t)^{1/2}}{\sqrt{\Delta_t}} \succeq \frac{\delta}{\sqrt{\Delta_t}}(\mathrm{C}_t^\top \mathrm{C}_t)^{1/2},
\end{aligned}
$$

where the second inequality is due to $\sqrt{\Delta_t} + x \ge \sqrt{\Delta_t + x^2}$ for $x \ge 0$ and the third inequality is due to Lemma 4.

And in the other case $\delta \geq \sqrt{\Delta_t}$, we have

$$\begin{aligned}\mathrm{H}_t &= \delta \mathrm{I}_d + (\mathrm{S}_t^\top \mathrm{S}_t)^{1/2} \succeq \sqrt{\Delta_t}\mathrm{I}_d + (\mathrm{S}_t^\top \mathrm{S}_t)^{1/2} \\ &\succeq (\Delta_t \mathrm{I}_d + \mathrm{S}_t^\top \mathrm{S}_t)^{1/2} \succeq (\mathrm{C}_t^\top \mathrm{C}_t)^{1/2}.\end{aligned}$$

Thus, for any $\delta > 0$, we have

$$\mathrm{H}_t \succeq \min\left(1, \frac{\delta}{\sqrt{\Delta_t}}\right)(\mathrm{C}_t^\top \mathrm{C}_t)^{1/2}.$$

Then, we have

$$\begin{aligned}\sum_{t=1}^{T} \|f_t'(\boldsymbol{\beta}_t)\|_{\Psi_t^*}^2 &= \sum_{t=1}^{T}\left\langle \mathbf{g}_t, (\mathrm{H}_t)^{-1}\mathbf{g}_t\right\rangle \\ &\leq \sum_{t=1}^{T} \max\left(1, \frac{\sqrt{\Delta_t}}{\delta}\right)\left\langle \mathbf{g}_t, (\mathrm{G}_t^\dagger)^{1/2}\mathbf{g}_t\right\rangle \\ &\leq \max\left(1, \frac{\sqrt{\Delta_T}}{\delta}\right)\sum_{t=1}^{T}\left\langle \mathbf{g}_t, (\mathrm{G}_t^\dagger)^{1/2}\mathbf{g}_t\right\rangle \\ &\leq 2\max\left(1, \frac{\sqrt{\Delta_T}}{\delta}\right)\mathrm{tr}(\mathrm{G}_T^{1/2})\end{aligned} \tag{20}$$

where the last inequality is due to Lemma 3.

By substituting (19) and (20) into Lemma 2, we complete the proof.

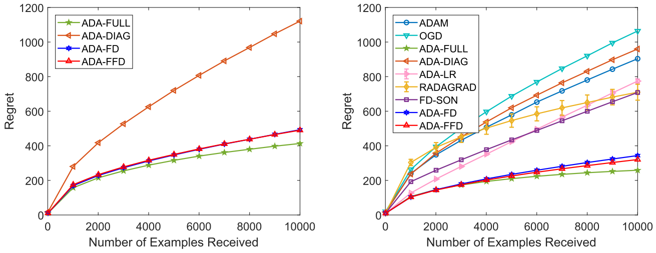
5 Experiments

In this section, we perform numerical experiments on online regression, online classification, training CNN, and fine-tuning much deeper CNN to verify the efficiency and effectiveness of our algorithms.

5.1 Baselines and the Default Setup

First, we show that ADA-FD and ADA-FFD approximate ADA-FULL well, and outperform ADA-DIAG. Second, we compare our methods with existing approximations of ADA-FULL including ADA-LR and RADAGRAD. Third, we compare our methods with FD-SON [30] that is an approximation of ONS by frequent directions. Fourth, we compare our algorithms with the classical online gradient descent (OGD) [44] which updates as $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \frac{\eta}{\sqrt{t}}\mathbf{g}_t$ and ADAM [45] that is a popular method for training neural networks. Note that ADAM have four parameters including the step size η, a constant δ, and two decay rates β_1, β_2.

In all experiments, by default, we search the step size η for all algorithms from the set of $\{1e{-}4, 1e{-}3, \ldots, 1\}$, and choose the value that achieves the best regret for experiments on online regression or the best training performance for other experiments. To simplify the parameter settings and control the computational cost, we simply set $\tau = 20$ for algorithms using matrix approximation by default. Moreover, we set $\delta = 1e{-}8, \beta_1 = 0.9, \beta_2 = 0.999$ for ADAM as recommended by Kingma and Ba [45], and also set $\delta = 1e{-}8$ for ADA-DIAG. For ADA-FULL, ADA-LR, RADAGRAD, and FD-SON, we select the parameter δ from the set of $\{1e{-}4, 1e{-}3, \ldots, 10\}$, and choose the value that achieves the best regret for experiments on online regression or the best training performance for other experiments. For our ADA-FD and ADA-FFD with the default $\tau = 20$, we set $\delta = 1$ when combining with the composite mirror descent (CMD) method, and set $\delta = 10$ when combining with the primal-dual subgradient (PDS) method.



(a) Regret for PDS (b) Regret for CMD

Fig. 1. The comparison of regret among different algorithms with the default $\tau = 20$ on synthetic data for the PDS method and CMD method.

In the above way, by default, our ADA-FD and ADA-FFD only have one varying hyperparameter—the step size η in different experiments.

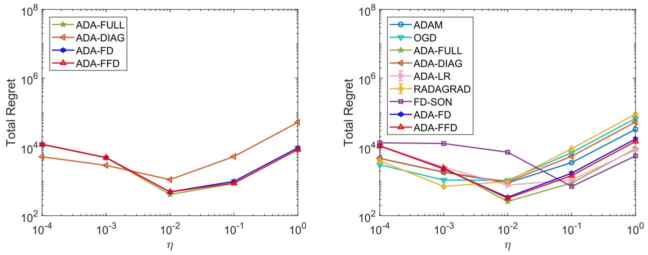
5.2 Online Regression

We first consider the problem of online regression where $f_t(\boldsymbol{\beta}) = |\boldsymbol{\beta}^\top \mathbf{x}_t - y_t|$ and $y_t = \boldsymbol{\beta}_*^\top \mathbf{x}_t$ with ideal synthetic data. Specifically, we set $T = 10{,}000$, $d = 500$, and $\boldsymbol{\beta}_* = \hat{\boldsymbol{\beta}}_*/\|\hat{\boldsymbol{\beta}}_*\|_2$, where each entry of $\hat{\boldsymbol{\beta}}_*$ is drawn independently from $\mathcal{N}(0,1)$. In order to meet the requirement of the approximately low-rank structure, each data point $\mathbf{x}_t$ is sampled independently from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ where $\boldsymbol{\mu} = \mathbf{1}$ and Σ has rapidly decaying eigenvalues $\lambda_j(\Sigma) = \lambda_0 j^{-\alpha}$ with $\alpha = 2$ and $\lambda_0 = 100$. For the PDS method, we compare ADA-FD and ADA-FFD against ADA-FULL and ADA-DIAG. And for the CMD method, we compare ADA-FD and ADA-FFD against all baselines. For fairness, all algorithms are run with the same permutation of the function sequence. Due to the randomness of ADA-LR and RADAGRAD, their experiments are run 20 times, and we report the average regret with standard deviations.

Fig. 1 shows the comparison of regret among different algorithms with the default $\tau = 20$. For both the PDS method and CMD method, our two algorithms outperform ADA-DIAG, and are close to ADA-FULL. For the CMD method, our two algorithms outperform ADA-LR, RADAGRAD, FD-SON, OGD, and ADAM. Note that according to the default setup, we need to search the best step size η from the set of $\{1e{-}4, 1e{-}3, \ldots, 1\}$ for all algorithms. So, it is reasonable to investigate whether our algorithms and other baselines are similarly sensitive to the value of η. To this end, we further present the comparison of the total regret among different algorithms with different η in Fig. 2. We find that all algorithms are similarly sensitive to the value of η. So, it is fair to search the best value of η for each algorithm.

Fig. 3 shows the total regret and running time of all algorithms with $\tau = \{20, 40, 60, 80\}$ for the CMD method. Note that the parameter δ of our two algorithms with $\tau > 20$ is selected according to the same way used for ADA-FULL. From this figure, we have the following observations.

- When τ is increased from 20 to 80, the total regret of algorithms using matrix approximation is decreased, and our ADA-FD and ADA-FFD always outperform ADA-LR, RADAGRAD, and FD-SON. The running time of each algorithm using matrix approximation



(a) Total Regret for PDS (b) Total Regret for CMD

Fig. 2. The comparison of regret among different algorithms with different η on synthetic data for the PDS method and CMD method.

increases as τ becomes larger. ADA-LR is much slower than all other methods except ADA-FULL. Our ADA-FD is significantly faster than ADA-FULL and as fast as RADAGRAD and FD-SON. Moreover, ADA-FFD is faster than ADA-FD.

- With $\tau = 80$, our ADA-FD and ADA-FFD can slightly outperform ADA-FULL. It is reasonable, because our ADA-FD and ADA-FFD maintain a strictly low-rank approximation of gradient outer products, which could eliminate possible noises in the approximately low-rank data.

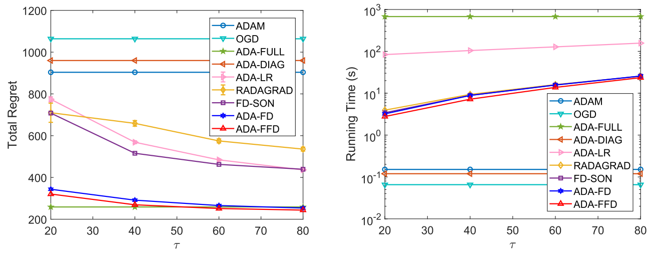
Furthermore, it is obvious that our ADA-FD with $\tau = 501$ and ADA-FFD with $\tau = 251$ will have the same regret as ADA-FULL.[1] Therefore, larger τ might not be better for minimizing the regret, which is clearly verified by Fig. 4. Moreover, from Fig. 4, we notice that the regret of ADA-FFD suddenly degrades back to that of ADA-FULL when τ changes from 250 to 251. It is reasonable, because the process of ADA-FFD with $\tau = 251$ significantly differs from that of ADA-FFD with $\tau = 250$. For ADA-FFD with $\tau = 251$, when t increases, the rank of $\mathrm{V}_t\mathrm{M}_t\mathrm{V}_t^\top$ first gradually increases from 1 to 500, and then is always kept at 500. However, with $\tau = 250$, every time the rank of $\mathrm{V}_t\mathrm{M}_t\mathrm{V}_t^\top$ reaches $2\tau = 500$, the *if* statement in step 12 of Algorithm 4 is triggered. In such cases, the rank of $\mathrm{V}_t\mathrm{M}_t\mathrm{V}_t^\top$ will be reduced to $\tau - 1 = 249$, because $\tau + 1 = 251$ columns of the matrix $\mathrm{V}_t\mathrm{M}_t^{1/2}$ are zeroed out. So, for ADA-FFD with $\tau = 250$, there exist *cyclical changes* about the rank of $\mathrm{V}_t\mathrm{M}_t\mathrm{V}_t^\top$: it repeatedly grows from $\tau - 1$ to 2τ.

Since the regret of ADA-FD is close to that of ADA-FFD, to facilitate presentations, we only report the results of ADA-FFD in the following.

5.3 Online Classification

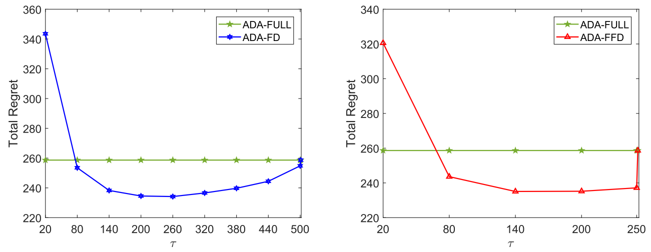
Then, we perform online classification to evaluate the performance of our ADA-FFD with two real world datasets from LIBSVM repository [46]: Gisette and Epsilon which are high-dimensional and dense. At each round, the learning algorithm receives a single example $(\mathbf{x}_t, y_t)$ and suffers the squared hinge loss $f_t(\boldsymbol{\beta}) = \left(\max\left(0, 1 - y_t\boldsymbol{\beta}^\top\mathbf{x}_t\right)\right)^2$. Each algorithm ends with a single pass through the training data. Following Duchi *et al.* [26], we adopt two metrics to measure the performance: the online mistakes and the offline accuracy on the testing data. For the PDS method, we compare ADA-FFD against ADA-DIAG. And for the CMD method, we compare ADA-FFD against ADA-DIAG, RADAGRAD, FD-SON, OGD, and ADAM. We omit the results of ADA-FULL and ADA-LR, because they are too slow. Both datasets are divided into the training part and testing part, and the numbers of training and testing examples are shown in Table 1.

For both datasets, we repeat the experiments of all algorithms 20 times with random permutations of training data, and report the average mistakes and test accuracy with standard deviations. Figs. 5 and 6 show the comparison of mistakes and test accuracy among different algorithms. We find that our ADA-FFD outperforms ADA-DIAG, RADAGRAD, FD-SON, OGD, and ADAM in terms of mistakes on Gisette and test accuracy on both datasets. In terms of mistakes on Epsilon, our ADA-FFD outperforms ADA-DIAG, RADAGRAD, OGD, and ADAM. Fig. 7 shows the comparison of running time among different algorithms. Our ADA-FFD is faster than RADAGRAD and FD-SON when $d = 5000$ and $d = 2000$. Compared with Fig. 3b, the advantage of our ADA-FFD in running time is more obvious for the larger d.



(a) Total Regret (b) Running Time

Fig. 3. The comparison of regret and running time among different algorithms with different τ on synthetic data for the CMD method.



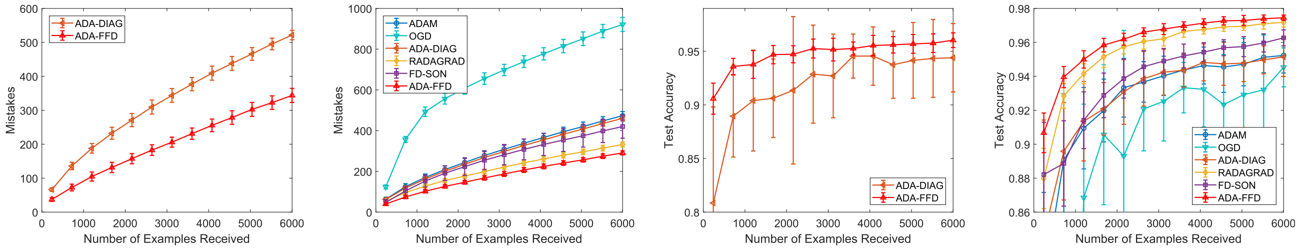
(a) ADA-FD vs. ADA-FULL (b) ADA-FFD vs. ADA-FULL

Fig. 4. The detailed comparison between ADA-FD and ADA-FFD with different τ and ADA-FULL on synthetic data for the CMD method.

TABLE 1
Datasets Used in Experiments

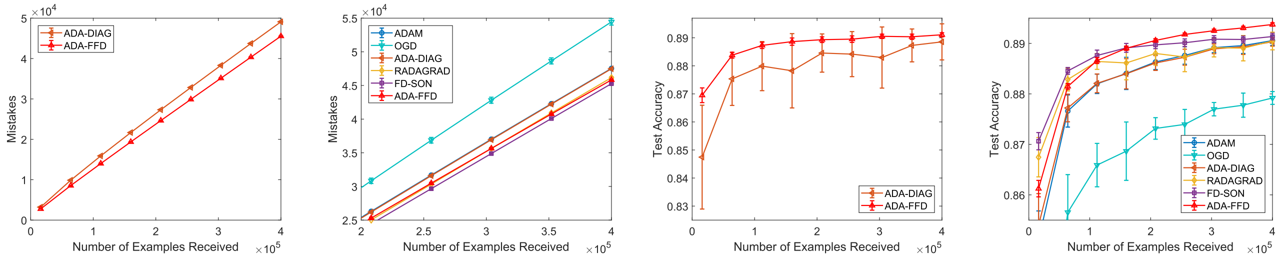
Dataset	#Examples	#Features	#Classes
Gisette	6,000/1,000	5,000	2
Epsilon	400,000/100,000	2,000	2
MNIST	60,000/10,000	784	10
CIFAR10	50,000/10,000	3,072	10
CIFAR100	50,000/10,000	3,072	100
SVHN	73,257/26,032	3,072	10

[1] ADA-FFD with $\tau = 251$ is equivalent to ADA-FULL for $d = 500$, since the *if* statement at step 12 of Algorithm 4 will never be triggered.



(a) Mistakes for PDS (b) Mistakes for CMD (c) Test Accuracy for PDS (d) Test Accuracy for CMD

Fig. 5. The comparison of mistakes and test accuracy among different algorithms on Gisette for the PDS method and CMD method.



(a) Mistakes for PDS (b) Mistakes for CMD (c) Test Accuracy for PDS (d) Test Accuracy for CMD

Fig. 6. The comparison of mistakes and test accuracy among different algorithms on Epsilon for the PDS method and CMD method.

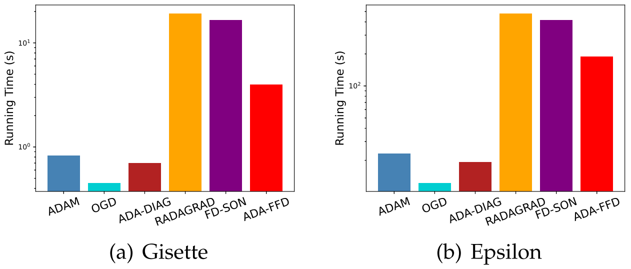
5.4 Non-Convex Optimization in CNN

Besides convex optimization, ADA-DIAG incorporated into the CMD method has also been applied to non-convex optimization such as training neural networks and has performed well. Therefore, we take convolutional neural networks as an example and compare ADA-FFD against ADA-DIAG, RADAGRAD, FD-SON, OGD, and ADAM on training classical neural networks. We consider a 4-layer CNN from the Keras examples directory,[2] which contains two 3×3 convolutional layers and two fully connected layers. The first convolutional layer generates 32 channels, and the second convolutional layer generates 64 channels followed by 2×2 max-pooling and 0.25 dropout. The first fully connected layer contains 128 hidden units followed by 0.5 dropout, and the second fully connected layer is a logistic layer. The activation functions of all layers except the logistic layer are ReLU, and the objective is the cross-entropy loss. We use this simple and standard architecture to perform classification on the MNIST [47], CIFAR10 [48], CIFAR100 [48], and SVHN [49] datasets.

Following Krummenacher *et al.* [27], we only consider applying ADA-FFD, RADAGRAD, and FD-SON to the convolutional layers, and other layers are still trained with ADA-DIAG. For all algorithms, we run 20 times with the batch size 128, and report the average training loss and test accuracy with standard deviations. Fig. 8 shows the comparison of training loss and test accuracy during training among different algorithms. We find that our ADA-FFD outperforms ADA-DIAG, RADAGRAD, FD-SON, and OGD on all datasets when applied to training CNN. Moreover, our ADA-FFD outperforms ADAM on CIFAR 10 and CIFAR100, and is close to ADAM on MNIST and SVHN. We also note that RADAGRAD is outperformed by ADA-DIAG when the same running time is used, which validates that our ADA-FFD has a better ability to approximate ADA-FULL than RADAGRAD. To make a clear comparison about the time complexity, Fig. 9 shows the comparison of running time costed by one epoch of each algorithm. We find that even if all algorithms run with the same number of epochs, our ADA-FFD is only a little slower than ADA-DIAG, OGD, and ADAM.

5.5 Fine-Tuning Much Deeper CNN

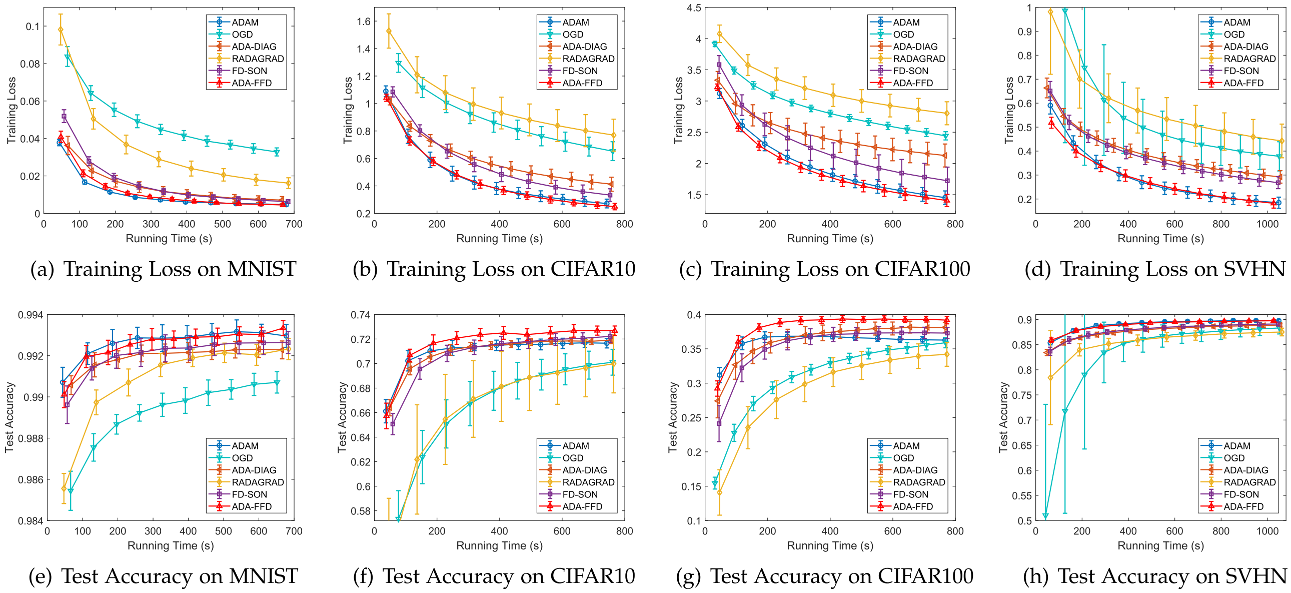
We also consider the problem of fine-tuning VGG19 [50] to adapt for CIFAR10, which contains sixteen convolutional layers and three fully connected layers. Note that VGG19 is originally designed for ImageNet [51] with 1,000 classes, while CIFAR10 only has 10 classes. Therefore, we replace the three fully connected layers of VGG19 with one fully connected layer that contains 10 hidden units and adopts softmax as the activation function. Furthermore, because 32×32 images in CIFAR10 are invalid inputs for VGG19, we resize all images in CIFAR10 to 64×64. Before fine-tuning the modified VGG19, we initialize its convolutional layers with the weights of VGG19 pre-trained on



(a) Gisette (b) Epsilon

Fig. 7. The comparison of running time among different algorithms on Gisette and Epsilon for the CMD method.

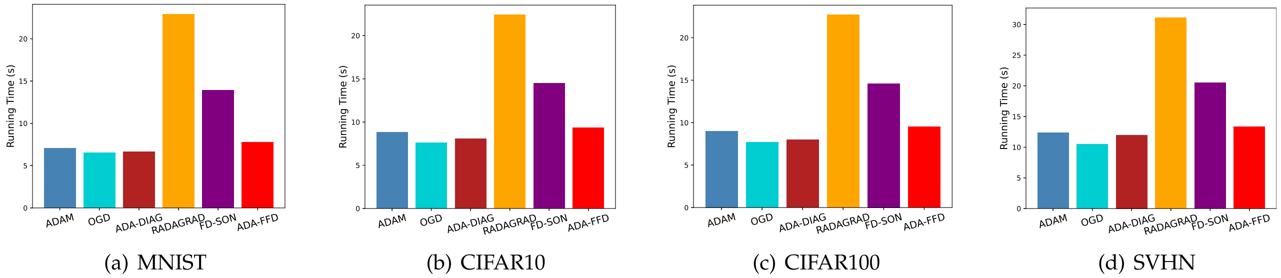
[2] https://github.com/keras-team/keras/tree/2.1.2/examples



(a) Training Loss on MNIST (b) Training Loss on CIFAR10 (c) Training Loss on CIFAR100 (d) Training Loss on SVHN

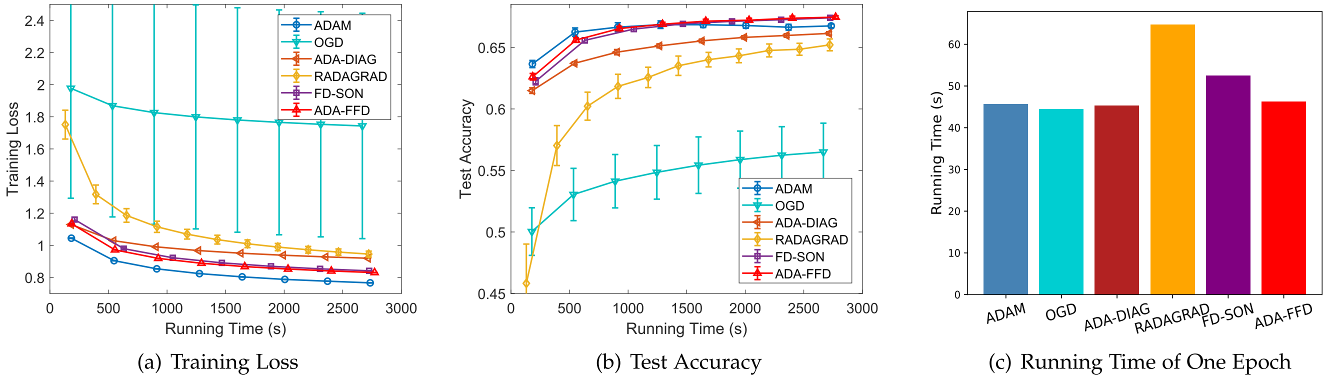
(e) Test Accuracy on MNIST (f) Test Accuracy on CIFAR10 (g) Test Accuracy on CIFAR100 (h) Test Accuracy on SVHN

Fig. 8. The comparison of training loss and test accuracy among different algorithms on training CNN.



(a) MNIST (b) CIFAR10 (c) CIFAR100 (d) SVHN

Fig. 9. The comparison of running time costed by one epoch of each algorithm on training CNN.



(a) Training Loss (b) Test Accuracy (c) Running Time of One Epoch

Fig. 10. The comparison of training loss, test accuracy, and running time of one epoch among different algorithms on fine-tuning VGG19 for CIFAR10.

ImageNet.[3] Then, we freeze the weights of all convolutional layers, and only train the top fully connected layer using the resized CIFAR10.

For all algorithms, we run 20 times with the batch size 128, and report the average training loss and test accuracy with standard deviations. Fig. 10 shows the comparison of training loss, test accuracy, and running time of one epoch among different algorithms on fine-tuning VGG19 for CIFAR10. In the aspects of training loss and test accuracy, we find that our ADA-FFD outperforms ADA-DIAG, RADAGRAD, and OGD, and is slightly better than FD-SON. Although the training loss of our ADA-FFD is worse than that of ADAM, the test accuracy of ADA-FFD is better.

[3] https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5

Moreover, if all algorithms run with the same number of epochs, our ADA-FFD is only a little slower than ADA-DIAG, OGD, and ADAM.

6 Conclusion

In this paper, we first present ADA-FD to approximate ADA-FULL using frequent directions, which reduces the time and space complexities to $O(\tau^2 d)$ and $O(\tau d)$, respectively, and has the similar efficiency compared to ADA-DIAG. Then, we further present ADA-FFD to reduce the average time complexity to $O(\tau d)$ that is linear in both the dimensionality d and the sketching size τ, which only needs to double the space complexity of ADA-FD for accelerating frequent directions used in it. Furthermore, according to our theoretical analysis, our ADA-FD and ADA-FFD enjoy the regret bound close to that of ADA-FULL when the outer product matrix of gradients is approximately low-rank. Numerical experiments on different problems and datasets demonstrate the efficiency and effectiveness of our algorithms.

Acknowledgments

This work was supported in part by the NSFC under Grant 61976112, in part by the NSFC-NRF Joint Research Project under Grant 61861146001, and JiangsuSF under Grant BK20200064. A preliminary version of this paper was presented at the 27th International Joint Conference on Artificial Intelligence in 2018 [1].

References

[1] Y. Wan, N. Wei, and L. Zhang, "Efficient adaptive online learning via frequent directions," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2748–2754.

[2] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[3] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, 1999.

[4] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari, "Efficient bandit algorithms for online multiclass prediction," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 440–447.

[5] L. Zhang, R. Jin, C. Chen, J. Bu, and X. He, "Efficient online learning for large-scale sparse kernel logistic regression," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1219–1225.

[6] L. Zhang, J. Yi, R. Jin, M. Lin, and X. He, "Online kernel learning with a near optimal sparsity bound," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 621–629.

[7] L. Cao, F. Sun, H. Li, and W. Huang, "Advancing the incremental fusion of robotic sensory features using online multi-kernel extreme learning machine," *Front. Comput. Sci.*, vol. 11, no. 2, pp. 276–289, 2017.

[8] H. B. McMahan *et al.*, "Ad click prediction: A view from the trenches," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 1222–1230.

[9] D. Agarwal, B. Long, J. Traupman, D. Xin, and L. Zhang, "LASER: A scalable response prediction platform for online advertising," in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, 2014, pp. 173–182.

[10] L. Zhang, T. Yang, R. Jin, Y. Xiao, and Z.-H. Zhou, "Online stochastic linear optimization under one-bit feedback," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 392–401.

[11] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman, "Online metric learning and fast similarity search," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 761–768.

[12] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *J. Mach. Learn. Res.*, vol. 11, pp. 1109–1135, 2010.

[13] G. Tsagkatakis and A. Savakis, "Online distance metric learning for object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 12, pp. 1810–1821, Dec. 2011.

[14] X. Gao, S. C. Hoi, Y. Zhang, J. Wan, and J. Li, "SOML: Sparse online metric learning with application to image retrieval," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1206–1212.

[15] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, 2010.

[16] B. Lois and N. Vaswani, "Online matrix completion and online robust PCA," in *Proc. IEEE Int. Symp. Inf. Theory*, 2015, pp. 1826–1830.

[17] C. Jin, S. M. Kakade, and P. Netrapalli, "Provable efficient online matrix completion via non-convex stochastic gradient descent," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4520–4528.

[18] R. Zhao and V. Y. F. Tan, "Online nonnegative matrix factorization with outliers," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 2662–2666.

[19] W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou, "One-pass AUC optimization," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 906–914.

[20] R. Busa-Fekete, B. Szörényi, K. Dembczynski, and E. Hüllermeier, "Online F-measure optimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 595–603.

[21] Y. Ying, L. Wen, and S. Lyu, "Stochastic online AUC maximization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 451–459.

[22] L. Zhang, S. Lu, and Z.-H. Zhou, "Adaptive online learning in dynamic environments," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 1323–1333.

[23] Y. Wan, W.-W. Tu, and L. Zhang, "Projection-free distributed online convex optimization with $O(\sqrt{T})$ communication complexity," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 9818–9828.

[24] R. Laxhammar and G. Falkman, "Online learning and sequential anomaly detection in trajectories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1158–1173, Jun. 2014.

[25] Y. Yuan, J. Fang, and Q. Wang, "Online anomaly detection in crowd scenes via structure analysis," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 548–561, Mar. 2015.

[26] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.

[27] G. Krummenacher, B. McWilliams, Y. Kilcher, J. M. Buhmann, and N. Meinshausen, "Scalable adaptive stochastic optimization using random projections," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1750–1758.

[28] Y. Wan and L. Zhang, "Accelerating adaptive online learning by matrix approximation," in *Proc. 22nd Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2018, pp. 405–417.

[29] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Mach. Learn.*, vol. 69, no. 2, pp. 169–192, 2007.

[30] H. Luo, A. Agarwal, N. Cesa-Bianchi, and J. Langford, "Efficient second order online learning by sketching," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 902–910.

[31] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff, "Frequent directions: Simple and deterministic matrix sketching," *SIAM J. Comput.*, vol. 45, no. 5, pp. 1762–1792, 2016.

[32] L. Xiao, "Dual averaging method for regularized stochastic learning and online optimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 2116–2124.

[33] J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari, "Composite objective mirror descent," in *Proc. 23rd Annu. Conf. Learn. Theory*, 2010, pp. 14–26.

[34] N. Nalko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011.

[35] E. Liberty, "Simple and deterministic matrix sketching," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 581–588.

[36] M. Ghashami and J. M. Phillips, "Relative errors for deterministic low-rank matrix approximations," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2014, pp. 707–717.

[37] S. Kaski, "Dimensionality reduction by random mapping: Fast similarity computation for clustering," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1998, pp. 413–418.

[38] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theor. Comput. Sci.*, vol. 312, no. 1, pp. 3–15, 2004.

[39] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *Found. Trends Mach. Learn.*, vol. 10, no. 1/2, pp. 1–157, 2014.

[40] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication," *SIAM J. Comput.*, vol. 36, no. 1, pp. 132–157, 2006.

[41] J. Misra and D. Gries, "Finding repeated elements," *Sci. Comput. Program.*, vol. 2, no. 2, pp. 143–152, 1982.

[42] W. W. Hager, "Updating the inverse of a matrix," *SIAM Rev.*, vol. 31, no. 2, pp. 221–239, 1989.

[43] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra Appl.*, vol. 415, no. 1, pp. 20–30, 2006.

[44] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 928–936.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[46] C.-C. Chang and C.-J. Lin, "LIBSVM : A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[47] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[48] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[49] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9. [Online]. Available: https://deeplearningworkshopnips2011.files.wordpress.com/2011/12/12.pdf

[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–14.

[51] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

Yuanyu Wan received the BS degree in software engineering from Sun Yat-Sen University, Guangzhou, China, in 2016, and the MS degree in computer science and technology from Nanjing University, Nanjing, China, in 2019. He is currently working toward the PhD degree with the Department of Computer Science and Technology, Nanjing University, Nanjing, China. His research interests include machine learning and optimization.

Lijun Zhang (Member, IEEE) received the BS and PhD degrees in software engineering and computer science from Zhejiang University, Hangzhou, China, in 2007 and 2012, respectively. He is currently a research professor with the School of Artificial Intelligence, Nanjing University, China. Prior to joining Nanjing University, he was a postdoctoral researcher with the Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan. His research interests include machine learning, optimization, information retrieval, and data mining.