Minimizing Adaptive Regret with One Gradient per Iteration

Guanghui Wang, Dakuan Zhao, Lijun Zhang
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
wanggh@lamda.nju.edu.cn, zdk@smail.nju.edu.cn, zhanglj@lamda.nju.edu.cn

Abstract

To cope with non-stationary environments, recent advances in online optimization have introduced the notion of *adaptive* regret, which measures the performance of an online learner against different comparators within different time intervals. Previous studies have proposed various algorithms to yield low adaptive regret under different scenarios. However, all of existing algorithms need to query the gradient of the loss function at least $O(\log t)$ times in every iteration t, which hinders their applications to broad domains, especially when the evaluation of gradients is expensive. To address this limitation, we propose a series of computationally efficient algorithms for minimizing the adaptive regret of general convex, strongly convex and exponentially concave functions respectively. The key idea is to replace each loss function with a carefully designed *surrogate loss*, which bounds the original loss function from below. We show that the proposed algorithms only query the gradient *once* per iteration, and attain the same theoretical guarantees as previous optimal algorithms. Empirical results demonstrate the efficiency and effectiveness of our methods.

1 Introduction

Online convex optimization is a powerful framework for modeling sequential decision making [Hazan, 2016]. It can be deemed as a repeated game where a learner competes against an adversary: In every iteration t, firstly the learner selects an action $\mathbf{x}_t$ from a convex set $\mathcal{D} \subseteq \mathbb{R}^d$, at the same time, the adversary reveals a convex function $f_t(\cdot) : \mathcal{D} \mapsto \mathbb{R}$, and the learner incurs a loss $f_t(\mathbf{x}_t)$. The goal is to minimize the cumulative loss over T iterations. The standard performance metric is *regret*, defined as the difference between the cumulative loss of the learner and that of the optimal action in hindsight [Shalev-Shwartz, 2012]:

$$\mathrm{R}_{f_1,\ldots,f_T} = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{D}} \sum_{t=1}^{T} f_t(\mathbf{x}),$$

which is typically referred to as *static* regret since the comparator is time-invariant. Despite that static regret is a reasonable benchmark in many contexts and has led to plentiful online algorithms [Hazan, 2016], it fails to capture the hardness of real-world applications where the environments are non-stationary and no single fixed action performs well in general (e.g., portfolio management, ad recommendation). As a consequence, a more stringent performance metric termed adaptive regret [Hazan and Seshadhri, 2007] was proposed and received significant interest recently. Given a parameter τ, the **S**trongly version of **A**daptive **R**egret (SAR) is defined as the algorithm's maximum static regret over any time interval of length τ [Daniely *et al.*, 2015]:

$$\mathrm{SAR}_{f_1,\ldots,f_T}(\tau) = \max_{[q,q+\tau-1]\subseteq[T]} \left\{ \mathrm{R}_{f_q,\ldots,f_{q+\tau-1}} \right\}, \tag{1}$$

where $[T]$ is the shorthand of $\{1, ..., T\}$. Minimizing SAR forces the algorithm to keep a low static regret in any time interval of length τ.

In the past decade, a wide variety of algorithms that guarantee low SAR have been proposed for different types of convex functions [Hazan and Seshadhri, 2007; Hazan and Seshadhri, 2009; Daniely *et al.*, 2015; Jun *et al.*, 2017; Zhang *et al.*, 2017]. However, different from traditional algorithms under the static setting that only query the gradient once in each iteration, existing approaches under the adaptive setting query the gradient at least $O(\log t)$ times in the t-th round. The price paid for adaptivity impedes them to many applications, especially when the evaluation of gradients is expensive. For example, in nuclear norm minimization [Ji and Ye, 2009], we have to perform SVD to compute the gradient, which is a costly step for large-scale matrices; another example is the mini-batch optimization [Li *et al.*, 2014], where the gradient is obtained by averaging the gradients of all samples inside a batch.

Contribution. In this paper, we propose a series of algorithms for minimizing adaptive regret that only query the gradient once in each iteration. Similar to previous approaches for adaptive regret, our algorithms follow the Learning with Expert Advice (LEA) framework [Cesa-Bianchi and Lugosi, 2006]. The basic idea is to maintain many algorithms for minimizing static regret as experts to output a set of actions, and employ a weighting method to combine these actions and select $\mathbf{x}_t$. In order to output an action, each expert needs to compute its own gradient. In contrast to previous algorithms where each expert queries the gradient of $f_t(\cdot)$ directly, the expert kept in our algorithm queries the gradient of a *surro-*

gate loss [van Erven and Koolen, 2016]. It can be considered as a conversion from the original learning problem to a new one, where the loss revealed by the adversary in each iteration is the surrogate loss, rather than $f_t(\cdot)$. The surrogate losses are carefully designed to satisfy the following properties.

- Their gradients can be computed efficiently by exploiting $\nabla f_t(\mathbf{x}_t)$, which is shared among all experts, *without further querying* $\nabla f_t(\cdot)$ *at different points*;
- The SAR of any algorithm on the new problem is an upper bound of that on the original problem.

Equipped with surrogate losses, our algorithms only need to query the gradient once in each round, and achieve $O(\sqrt{\tau \log T})$, $O(\log^2 T)$ and $O(d \log^2 T)$ SAR bounds for general convex, strongly convex and exponentially concave functions respectively. These upper bounds match the theoretical guarantees of the optimal SAR methods.

2 Related Work

In the literature, most studies are devoted to the minimization of static regret [Hazan, 2016]. For general convex and strongly convex functions, the classic algorithm Online Gradient Descent (OGD) enjoys $O(\sqrt{T})$ and $O(\log T)$ static regret bounds respectively [Zinkevich, 2003; Hazan *et al.*, 2007]. Both bounds are known to be minimax optimal [Abernethy *et al.*, 2009]. For exponentially concave functions, the state-of-the-art algorithm is Online Newton Step (ONS), which achieves an $O(d \log T)$ static regret bound [Hazan *et al.*, 2007].

To handle changing environments, the seminal work of Hazan and Seshadhri [2007] proposed *adaptive regret* (referred to as ***Weakly*** **A**daptive **R**egret, WAR), which is defined as the maximum static regret of an algorithm over any time interval:

$$\mathrm{WAR}_{f_1,\ldots,f_T} = \max_{[q,s]\subseteq[T]} \left\{ \mathrm{R}_{f_q,\ldots,f_s} \right\}.$$

To minimize this performance metric, Hazan and Seshadhri [2007] developed an algorithm called Following the Leading History (FLH), achieving $O(\sqrt{T \log T})$ and $O(d \log T)$ WAR for general convex and exponentially concave functions respectively. The essential idea of FLH is to maintain many low static regret algorithms (e.g., OGD or ONS) simultaneously as experts, and employ a *meta* algorithm to combine the outputs of experts and decide the final action. Each expert attains good static regret in a different time interval, and the meta algorithm can track the best expert by dynamically assigning weights to different experts according to their historical performances. Specifically, in the t-th round, FLH maintains t low static regret algorithms (OGD or ONS), and each algorithm needs to query the gradient of $f_t(\cdot)$ once. Thus, the number of gradient evaluations in FLH is linear with respect to t. To reduce the computational cost, Hazan and Seshadhri [2007] then proposed an enhanced version of FLH named Advanced Following the Leading History (AFLH). By removing some old experts according to a data streaming method, AFLH reduces the number of experts and gradient queries down to $O(\log t)$ per round, at a cost of an additional $\log T$ in its regret.

A major drawback of WAR is that it does not respect short time intervals well. For instance, an $O(\sqrt{T})$ type of static regret bound is meaningless for intervals of length $O(\sqrt{T})$. To address this limitation, Daniely *et al.* [2015] proposed strongly adaptive regret, which is defined in (1). Furthermore, Daniely *et al.* [2015] introduced the following definition:

Definition 1. *For any static learning problem, let $R(\tau)$ be its minimax static regret bound over τ iterations. An algorithm is said to be* strongly adaptive*, if*

$$\forall \tau > 0, \mathrm{SAR}_{f_1,\ldots,f_T}(\tau) = O(R(\tau)\mathrm{poly}(\log T)).$$

It can be proved that AFLH algorithm in Hazan and Seshadhri [2007] is in fact strongly adaptive for strongly convex and exponentially concave functions, but not for general convex functions. Daniely *et al.* [2015] then developed a strongly adaptive algorithm named Strongly Adaptive Online Learner (SAOL), which improves AFLH by introducing a new meta algorithm as well as a new expert removing policy. This algorithm achieves $O(\sqrt{\tau} \log T)$ SAR for general convex functions with $O(\log t)$ experts and gradient queries in round t. Later, Jun *et al.* [2017] proposed an algorithm named Coin Betting for Changing Environment (CBCE) that improves the SAR for convex functions to $O(\sqrt{\tau \log T})$ with $O(\log t)$ experts and gradient queries as well.

From the above discussions, we observe that all the previous methods for adaptive regret need to query the gradient at least $O(\log t)$ times in the t-th round. Our goal is to reduce the number of gradient evaluation to 1 per iteration.

3 Algorithms

In this section, we introduce our efficient low SAR algorithms for exponentially concave, strongly convex, as well as general convex functions respectively, and present their theoretical guarantees. We put the detailed analysis in the supplementary material due to the limitation of space.

Through out the paper, we use $\|\cdot\|$ to denote ℓ_2 norm, and $\Pi_{\mathcal{D}}^{A}(\cdot)$ the projection induced by a positive semi-definite matrix A, i.e.,

$$\Pi_{\mathcal{D}}^{A}(\mathbf{y}) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{D}} (\mathbf{y}-\mathbf{x})^\top A(\mathbf{y}-\mathbf{x}).$$

Before proceeding to specific algorithms, following previous studies [Hazan and Seshadhri, 2007; Jun *et al.*, 2017], we introduce the following assumptions and definitions [Boyd and Vandenberghe, 2004]:

Assumption 1. *The gradients of all loss functions are bounded by G, i.e., $\max_{\mathbf{x}\in\mathcal{D}} \|\nabla f_t(\mathbf{x})\| \le G$ for all t.*

Assumption 2. *The diameter of the decision set is bounded by D, i.e., $\max_{\mathbf{x}_1,\mathbf{x}_2\in\mathcal{D}} \|\mathbf{x}_1-\mathbf{x}_2\| \le D$.*

Definition 2. *A function $f : \mathcal{D} \mapsto \mathbb{R}$ is convex if $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$,*

$$f(\mathbf{x}_1) - f(\mathbf{x}_2) \le \nabla f(\mathbf{x}_1)^\top (\mathbf{x}_1 - \mathbf{x}_2).$$

Definition 3. *A function $f : \mathcal{D} \mapsto \mathbb{R}$ is λ-strongly convex if $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$,*

$$f(\mathbf{x}_1) - f(\mathbf{x}_2) \le \nabla f(\mathbf{x}_1)^\top (\mathbf{x}_1 - \mathbf{x}_2) + \frac{\lambda}{2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

Algorithm 1 The meta algorithm for exp-concave and strongly convex functions

```
1: Initialize $\mathcal{S}_1 = \{E_1\}$, $q_1 = 1$
2: for $t = 1, ..., T$ do
3:     for $E_i \in \mathcal{S}_t$ do
4:         if $q_i \neq t$ then
5:             Pass $\nabla f_{t-1}(\mathbf{x}_{t-1})$ to expert $E_i$
6:         end if
7:         Get action $\mathbf{x}_{i,t}$ from expert $E_i$
8:     end for
9:     Choose $\mathbf{x}_t$ by (3)
10:    Observe $\nabla f_t(\mathbf{x}_t)$
11:    Remove experts whose $e_i$ are less than $t$
12:    $\hat{n} = |\mathcal{S}_t| + 1$
13:    Initialize $E_{\hat{n}}$, set $p_{\hat{n},t+1} = \frac{1}{t+1}$, $q_{\hat{n}} = t$, compute $e_{\hat{n}}$
14:    $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{E_{\hat{n}}\}$
15:    for $E_i \in \mathcal{S}_{t+1}$ and $q_i \neq t$ do
16:        $\hat{p}_{i,t+1} = \begin{cases} p_{i,t}\exp(-\alpha^{ec} L_t^{ec}(\mathbf{x}_{i,t})), & \textit{exp-concave} \\ p_{i,t}\exp(-\alpha^{sc} L_t^{sc}(\mathbf{x}_{i,t})), & \textit{strongly convex} \end{cases}$
17:    end for
18:    for $E_i \in \mathcal{S}_{t+1}$ and $q_i \neq t$ do
19:        $p_{i,t+1} = (1 - \frac{1}{t+1}) \frac{\hat{p}_{i,t+1}}{\sum_{E_j \in \mathcal{S}_{t+1}, q_j \neq t} \hat{p}_{j,t+1}}$
20:    end for
21: end for
```

Algorithm 2 The algorithm for expert E_i (exp-concave version)

```
1: if $q_i = t$ then
2:     $A_{i,t} = \epsilon I_d$, $\mathbf{x}_{i,t} = \mathbf{0}$
3: else
4:     $A_{i,t} = A_{i,t-1} + \nabla L_{t-1}^{ec}(\mathbf{x}_{i,t-1}) \nabla L_{t-1}^{ec}(\mathbf{x}_{i,t-1})^\top$
5:     $\mathbf{y}_{i,t} = \mathbf{x}_{i,t-1} - \frac{1}{\gamma^{ec}} A_{i,t}^{-1} \nabla L_{t-1}^{ec}(\mathbf{x}_{i,t-1})$
6:     $\mathbf{x}_{i,t} = \Pi_{\mathcal{D}}^{A_{i,t}}(\mathbf{y}_{i,t})$
7: end if
8: Output $\mathbf{x}_{i,t}$
```

Definition 4. *A function $f : \mathcal{D} \mapsto \mathbb{R}$ is α-exponentially concave (abbreviated to α-exp-concave), if* $\exp(-\alpha f(\cdot))$ *is concave over domain $\mathcal{D}$.*

Finally, we introduce the following lemma for exp-concave functions [Hazan *et al.*, 2007]:

Lemma 1. *If Assumptions 1 and 2 hold, and $f_t : \mathcal{D} \mapsto \mathbb{R}$ is α-exp-concave, we have $\forall\, \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$,*

$$\begin{aligned} f_t(\mathbf{x}_1) - f_t(\mathbf{x}_2) \leq & \nabla f_t(\mathbf{x}_1)^\top (\mathbf{x}_1 - \mathbf{x}_2) \\ & - \frac{\gamma}{2} \left((\mathbf{x}_1 - \mathbf{x}_2)^\top \nabla f_t(\mathbf{x}_1) \right)^2, \end{aligned} \tag{2}$$

where $\gamma = \frac{1}{2}\min\{\frac{1}{4GD}, \alpha\}$.

3.1 Exp-Concave Functions

Our algorithm named MARSL-ec (**M**inimizing **A**daptive **R**egret with **S**urrogate **L**osses for **e**xp-**c**oncave functions) is a two-level hierarchical construction, summarized in Algorithm 1 (the meta algorithm) and Algorithm 2 (the expert algorithm). In each round t, we maintain a set of experts $\mathcal{S}_t = \{E_1, ..., E_n\}$. We use q_i to denote the round in which E_i is added to the expert set. At the beginning of every round, each expert $E_i \in \mathcal{S}_t$ runs the expert algorithm once to output an action $\mathbf{x}_{i,t}$, and passes it to the meta algorithm (Algorithm 1, steps 3-8). Since we are dealing with exp-concave functions, we choose ONS as the expert algorithm. Next, a linear weighting method is applied to combine the outputs of experts and select $\mathbf{x}_t$ (Algorithm 1, step 9):

$$\mathbf{x}_t = \sum_{E_i \in \mathcal{S}_t} p_{i,t} \mathbf{x}_{i,t}. \tag{3}$$

In step 11, following the data streaming rule in Hazan and Seshadhri [2007], we remove some experts from $\mathcal{S}_t$ to keep the number of experts $|\mathcal{S}_t| = O(\log t)$. Specifically, expert E_i will be removed in round e_i, defined as

$$e_i = t + 4 * 2^{u(q_i)} + 1, \tag{4}$$

where $u(q_i)$ is a number such that $2^{u(q_i)}$ is the largest power of 2 that divides q_i. After the experts are removed, we initialize a new expert $E_{\hat{n}}$ with $\hat{n} = |\mathcal{S}_t| + 1$, and add it to $\mathcal{S}_t$ to get $\mathcal{S}_{t+1}$ (steps 12-14). In steps 15-17, the weights of experts are updated by their own losses through the exponential weighting method. Finally, in steps 18-20, the weights are normalized to keep the sum to be 1.

The major difference between AFLH [Hazan and Seshadhri, 2007] and our algorithm is the loss function revealed to experts. While AFLH directly reveals $f_t(\cdot)$ to all experts in round t, we use a surrogate loss $L_t^{ec}(\cdot)$. Specifically, consider an expert $E_i \in \mathcal{S}_t$. As mentioned above, in round t, E_i has to perform ONS to output $\mathbf{x}_{i,t}$. Let $\mathbf{x}_{i,t-1}$ be its output in round $t-1$. In AFLH, $\mathbf{x}_{i,t}$ is computed as:

$$\mathbf{x}_{i,t} = \mathbf{x}_{i,t-1} - \frac{1}{\gamma} M_{i,t}^{-1} \nabla f_{t-1}(\mathbf{x}_{i,t-1}), \tag{5}$$

where

$$M_{i,t} = M_{i,t-1} + \nabla f_{t-1}(\mathbf{x}_{i,t-1}) \nabla f_{t-1}(\mathbf{x}_{i,t-1})^\top. \tag{6}$$

Clearly, to compute $\mathbf{x}_{i,t}$, expert E_i has to query $\nabla f_{t-1}(\cdot)$ at $\mathbf{x}_{i,t-1}$. Because $|\mathcal{S}_t| = O(\log t)$, it leads to an $O(\log t)$ number of gradient evaluations in round t. Inspired by Metagrad [van Erven and Koolen, 2016], we introduce surrogate loss to tackle this drawback. According to Lemma 1, we construct a lower bound for $f_t(\cdot)$, i.e., $\forall \mathbf{u} \in \mathcal{D}$,

$$\begin{aligned} f_t(\mathbf{u}) \geq & \frac{\gamma}{2} (\mathbf{x}_t - \mathbf{u})^\top \nabla_t \nabla_t^\top (\mathbf{x}_t - \mathbf{u}) \\ & - \nabla_t^\top (\mathbf{x}_t - \mathbf{u}) + f_t(\mathbf{x}_t), \end{aligned} \tag{7}$$

where $\nabla_t := \nabla f_t(\mathbf{x}_t)$. Then, we define

$$L_t^{ec}(\mathbf{u}) = \frac{\gamma}{2} (\mathbf{x}_t - \mathbf{u})^\top \nabla_t \nabla_t^\top (\mathbf{x}_t - \mathbf{u}) - \nabla_t^\top (\mathbf{x}_t - \mathbf{u}), \tag{8}$$

and *take $L_t^{ec}(\cdot)$ as the loss for all experts in $\mathcal{S}_t$.* $L_t^{ec}(\cdot)$ enjoys the following property:

Lemma 2. *If Assumptions 1 and 2 hold, $f_t(\cdot)$ is α-exp-concave, then $L_t^{ec}(\cdot)$ is α^{ec}-exp-concave, where $\alpha^{ec} = \gamma / \left(1 + 2\gamma DG + \gamma^2 D^2 G^2\right)$. Besides, we have $\forall\, \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{D}$,*

$$\begin{aligned} L_t^{ec}(\mathbf{u}_1) - L_t^{ec}(\mathbf{u}_2) \leq & \nabla L_t^{ec}(\mathbf{u}_1)^\top (\mathbf{u}_1 - \mathbf{u}_2) \\ & - \frac{\gamma^{ec}}{2} \left((\mathbf{u}_1 - \mathbf{u}_2)^\top \nabla L_t^{ec}(\mathbf{u}_1) \right)^2, \end{aligned}$$

Algorithm 3 The algorithm for expert E_i (strongly convex version)

```
1: if $q_i = t$ then
2:     $\mathbf{x}_{i,t} = \mathbf{0}$
3: else
4:     $\mathbf{y}_{i,t} = \mathbf{x}_{i,t-1} - \frac{1}{t-q_i}\nabla L^{sc}_{t-1}(\mathbf{x}_{i,t-1})$
5:     $\mathbf{x}_{i,t} = \prod_{\mathcal{D}}^{I_d}(\mathbf{y}_{i,t})$
6: end if
7: Output $\mathbf{x}_{i,t}$
```

where $\gamma^{ec} = \frac{1}{2}\min\{\frac{1}{4G^{ec}D}, \alpha^{ec}\}$, $G^{ec} = \frac{5}{4}GD$.

By Lemma 2, $L^{ec}_t(\cdot)$ is exp-concave, therefore ONS can still be applied. Specifically, for expert E_i, its updating rule can be rewritten as:

$$\mathbf{x}_{i,t} = \mathbf{x}_{i,t-1} - \frac{1}{\gamma^{ec}}A^{-1}_{i,t}\nabla L^{ec}_{t-1}(\mathbf{x}_{i,t-1}), \tag{9}$$

where

$$A_{i,t} = A_{i,t-1} + \nabla L^{ec}_{t-1}(\mathbf{x}_{i,t-1})\nabla L^{ec}_{i,t-1}(\mathbf{x}_{i,t-1})^\top. \tag{10}$$

Because

$$\nabla L^{ec}_{t-1}(\mathbf{x}_{i,t-1}) = \gamma\nabla_{t-1}\nabla^\top_{t-1}(\mathbf{x}_{i,t-1} - \mathbf{x}_{t-1}) + \nabla_{t-1}, \tag{11}$$

expert E_i only needs to use $\nabla_{t-1} = \nabla f_{t-1}(\mathbf{x}_{t-1})$, instead of querying $\nabla f_{t-1}(\mathbf{x}_{i,t-1})$. In this way, our algorithm only queries the gradient once in each round.

As we have replaced $f_t(\cdot)$ with $L^{ec}_t(\cdot)$, after the gradient is revealed, we use $L^{ec}_t(\mathbf{x}_{i,t})$ to update $\hat{p}_{i,t}$:

$$\hat{p}_{i,t+1} = p_{i,t}\exp(-\alpha^{ec}L^{ec}_t(\mathbf{x}_{i,t})). \tag{12}$$

For α-exp-concave functions, MARSL-ec achieves the following SAR bound, which matches the optimal result in Hazan and Seshadhri [2007] up to a constant factor:

Theorem 1. *Suppose Assumptions 1 and 2 hold and all functions $f_1, \ldots, f_T$ are α-exp-concave, MARSL-ec achieves*

$$\begin{aligned}&\mathrm{SAR}_{f_1,\ldots,f_T}(\tau)\\ \le&(\log T + 1)\left(5\left(\frac{1}{\alpha^{ec}} + \frac{5}{4}GD^2\right)d\log T + 1\right)\\ =&O\left(d\log^2 T\right).\end{aligned}$$

We note that the concept of surrogate loss was used by van Erven and Koolen [2016]. Their work aims to develop an universal algorithm that can automatically adapt to different type of loss functions under the static setting. In contrast, this paper aims to minimize the adaptive regret in the dynamic setting.

3.2 Strongly Convex Functions

For strongly convex functions, we proposed an algorithm named MARSL-sc (**M**inimizing **A**daptive **R**egret with **S**urrogate **L**osses for **s**trongly **c**onvex functions), which uses the same meta algorithm as MARSL-ec, but employs a different surrogate loss and expert algorithm (Algorithm 3). Before proceeding to the algorithm, firstly we introduce the following lemma [Hazan *et al.*, 2007], which implies that strongly convex function is also exp-concave when the gradient is bounded:

Lemma 3. *If $f_t : \mathcal{D} \to \mathbb{R}$ is λ-strongly convex and Assumption 1 holds, then $f_t(\cdot)$ is λ/G^2-exp-concave.*

By Lemma 3, we can directly apply MARSL-ec to this learning scenario. However, it will lead to a linear dependence on d in the upper bound (Theorem 1). As an alternative, we introduce a new surrogate loss which suits the problem better. Following the same spirit as in Section 3.1, we first give a lower bound for $f_t(\cdot)$ based on Definition 3, i.e., $\forall \mathbf{u} \in \mathcal{D}$,

$$f_t(\mathbf{u}) \ge \nabla^\top_t(\mathbf{u} - \mathbf{x}_t) + \frac{\lambda}{2}(\mathbf{x}_t - \mathbf{u})^\top(\mathbf{x}_t - \mathbf{u}) + f_t(\mathbf{x}_t). \tag{13}$$

Then, the surrogate loss $L^{sc}_t(\cdot)$ is defined as:

$$L^{sc}_t(\mathbf{u}) = -\left[\nabla^\top_t(\mathbf{x}_t - \mathbf{u}) - \frac{\lambda}{2}(\mathbf{x}_t - \mathbf{u})^\top(\mathbf{x}_t - \mathbf{u})\right],$$

and we use $L^{sc}_t(\cdot)$ in place of $f_t(\cdot)$ for all experts in round t. Because the Hessian matrix of $L^{sc}_t(\cdot)$ is positive definite with parameter λ, $L^{sc}_t(\cdot)$ is λ-strongly convex, and therefore we can adopt OGD for strongly convex functions as the expert algorithm. Specifically, each expert $E_i \in \mathcal{S}_t$ uses the following rule to compute $\mathbf{x}_{i,t}$:

$$\mathbf{x}_{i,t} = \mathbf{x}_{i,t-1} - \eta_{i,t}\nabla L^{sc}_{t-1}(\mathbf{x}_{i,t-1}), \tag{14}$$

where $\eta_{i,t} = 1/(t - q_i)$ is the step size, and

$$\nabla L^{sc}_{t-1}(\mathbf{x}_{i,t-1}) = \lambda(\mathbf{x}_{i,t-1} - \mathbf{x}_{t-1}) - \nabla_{t-1}. \tag{15}$$

Again, each expert can output its action by only using ∇_{t-1}, instead of querying $\nabla f_{t-1}(\mathbf{x}_{i,t-1})$. Correspondingly, we use $L^{sc}_t(\mathbf{x}_{i,t})$ to update $\hat{p}_{i,t}$ at each round t:

$$\hat{p}_{i,t+1} = p_{i,t}\exp\left(-\alpha^{sc}L^{sc}_t(\mathbf{x}_{i,t})\right), \tag{16}$$

where α^{sc} is the parameter of exp-convexity of $L^{sc}_t(\cdot)$. Since $L^{sc}_t(\cdot)$ is λ-strongly convex, and $\forall \mathbf{u} \in \mathcal{D}$,

$$\|L^{sc}(\mathbf{u})\| \le \lambda D + G,$$

Lemma 3 implies $L^{sc}_t(\cdot)$ is $\lambda/(\lambda D + G)^2$-exp-concave. Consequently, we set $\alpha^{sc} = \lambda/(\lambda D + G)^2$.

For λ-strongly convex functions, we prove a similar SAR bound as Theorem 1, but independent from d:

Theorem 2. *Suppose Assumptions 1 and 2 hold, and all functions $f_1, \ldots, f_T$ are λ-strongly convex, MARSL-sc achieves*

$$\begin{aligned}&\mathrm{SAR}_{f_1,\ldots,f_T}(\tau)\\ \le&(\log T + 1)\left(\frac{(\lambda D + G)^2}{2\lambda}(\log T + 1) + 1\right)\\ =&O\left(\log^2 T\right).\end{aligned}$$

3.3 General Convex Functions

For general convex functions, we can still employ Algorithm 1 as the meta algorithm, and take OGD for general convex functions as its subroutine. However, it will only lead to a weakly adaptive algorithm, as indicated in Section 2. Alternatively, we build our meta algorithm based on the Coin Betting for Changing Environment (CBCE) algorithm of Jun *et al.* [2017]. The algorithm, named **M**inimizing **A**daptive **R**egret with **S**urrogate **L**osses for **g**eneral **c**onvex functions (MARSL-gc), is summarized in Algorithm 4 (the

Algorithm 4 The meta algorithm for general convex functions

```
1: Initialize $\mathcal{S}_1 = \{E_1\}$, $q_i = 1$
2: for $t = 1, ..., T$ do
3:   for $E_i \in \mathcal{S}_t$ do
4:     if $q_i \neq t$ then
5:       Pass $\nabla f_{t-1}(\mathbf{x}_{t-1})$ to expert $E_i$
6:     end if
7:     Get action $\mathbf{x}_{i,t}$ of expert $E_i$ (Algorithm 5)
8:   end for
9:   Choose $\mathbf{x}_t$ by (3), observe $\nabla f_t(\mathbf{x}_t)$ and $f_t(\mathbf{x}_t)$
10:  Remove experts whose $e_i$ are less than $t$
11:  for $E_i \in \mathcal{S}_t$ do
12:    Compute $\widetilde{g}_{i,t}$ by (21)
13:  end for
14:  $\hat{n} = |\mathcal{S}_t| + 1$
15:  Initialize $E_{\hat{n}}$, set $q_{\hat{n}} = t$ and compute $e_{\hat{n}}$
16:  $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{E_n\}$
17:  for $E_i \in \mathcal{S}_{t+1}$ do
18:    Compute $w_{i,t+1}$ and $\hat{p}_{i,t+1}$ by (17) and (18)
19:  end for
20:  $\mathbf{p}_{t+1} = \begin{cases} \hat{\mathbf{p}}_{t+1}/\|\hat{\mathbf{p}}_{t+1}\|_1, & \|\hat{\mathbf{p}}_{t+1}\|_1 > 0 \\ [\pi_{E_i}]_{E_i \in \mathcal{S}_t}, & otherwise \end{cases}$
21: end for
```

Algorithm 5 The algorithm for expert E_i (general convex version)

```
1: if $q_i = t$ then
2:   $\mathbf{x}_{i,t} = \mathbf{0}$
3: else
4:   $\mathbf{y}_{i,t} = \mathbf{x}_{i,t-1} - \frac{1}{2GD\sqrt{t-q_i}} \nabla f_{t-1}(\mathbf{x}_{t-1})$
5:   $\mathbf{x}_{i,t} = \prod_{\mathcal{D}}^{I_d}(\mathbf{y}_{i,t})$
6: end if
7: Output $\mathbf{x}_{i,t}$
```

meta algorithm) and Algorithm 5 (the expert algorithm). Similar to AFLH, CBCE follows the LEA framework with a data streaming method. In each round t, the meta algorithm receives $\mathbf{x}_{i,t}$, $i = 1, ..., n$, and selects $\mathbf{x}_t$ as in (3). Next, old experts are removed from $\mathcal{S}_t$ according to the same data streaming method as in Section 3.1, and a new expert $E_{\hat{n}}$ is initialized and added to $\mathcal{S}_t$ to get $\mathcal{S}_{t+1}$. Finally, the weights of experts are updated and normalized. However, instead of applying exponential weighting scheme, CBCE employs Sleeping Coin Betting method [Blum, 1997; Orabona and Pál, 2016] as the weight updating policy. In round t, after $\mathcal{S}_{t+1}$ is obtained, $\hat{p}_{i,t+1}$ is obtained as:

$$\hat{p}_{i,t+1} = \pi_i \max\{w_{i,t+1}, 0\}, \tag{17}$$

where

$$w_{i,t+1} = \frac{\sum_{j=q_i}^{t} \widetilde{g}_{i,j}}{t - q_i + \delta} \left(1 + \sum_{j=q_i}^{t} \widetilde{g}_{i,j} w_{i,j} \right), \tag{18}$$

$\widetilde{g}_{i,t}$ is computed as:

$$\widetilde{g}_{i,t} = \begin{cases} f_t(\mathbf{x}_t) - f_t(\mathbf{x}_{i,t}), & w_{i,t} > 0 \\ \max\{f_t(\mathbf{x}_t) - f_t(\mathbf{x}_{i,t}), 0\}, & otherwise \end{cases}$$

and $\pi_i = 1/q_i^2(1 + \lfloor \log q_i \rfloor)$ is the prior of E_i.

Our algorithm and CBCE differ in the loss functions revealed to experts. By Definition 2, a first attempt is to define the surrogate loss as:

$$L_t^{gc}(\mathbf{u}) = -\nabla f_t(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{u}). \tag{19}$$

However, because CBCE requires the loss function to lie in [0,1], we scale and redefine the surrogate loss function as:

$$L_t^{gc}(\mathbf{u}) = -\nabla f_t(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{u})/(2GD) + 1/2. \tag{20}$$

We replace $f_t(\cdot)$ with (20) as the loss function in round t. It can be easily verified that $\forall \mathbf{u} \in \mathcal{D}$, $L_t^{gc}(\mathbf{u}) \in [0, 1]$. Note that the gradient of $L_t^{gc}(\mathbf{u})$ is $\nabla L_t^{gc}(\mathbf{u}) = \nabla_t/2GD$, which implies that every expert in our algorithm shares the same gradient, instead of computing their own gradients respectively. Correspondingly, the update rule of $\widetilde{g}_{i,t}$ is rewritten as

$$\widetilde{g}_{i,t} = \begin{cases} L_t^{gc}(\mathbf{x}_t) - L_t^{gc}(\mathbf{x}_{i,t}), & w_{i,t} > 0 \\ \max\{L_t^{gc}(\mathbf{x}_t) - L_t^{gc}(\mathbf{x}_{i,t}), 0\}, & otherwise. \end{cases} \tag{21}$$

For general convex functions, MARSL-gc achieves the following SAR bound, which is on the same order as the optimal result in Jun *et al.* [2017]:

Theorem 3. *Suppose Assumptions 1 and 2 hold and all functions $f_1, \dots, f_T$ are convex, MARSL-gc achieves*

$$\begin{aligned} \mathrm{SAR}_{f_1,\dots,f_T}(\tau) &\le 2GD\sqrt{\tau} \left(\frac{6D}{\sqrt{2}-1} + 8\sqrt{7\log T + 5} \right) \\ &= O\left(\sqrt{\tau \log T} \right). \end{aligned}$$

4 Experiments

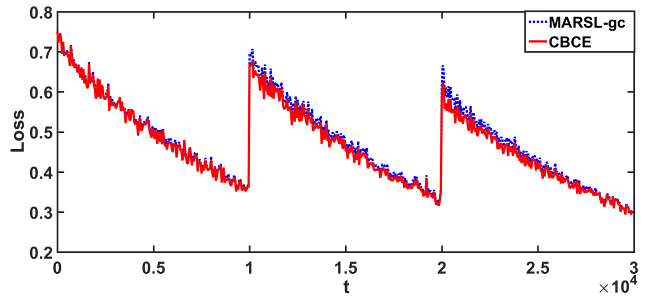
In this section, we present empirical results on different data sets to evaluate the proposed algorithms.

4.1 Synthetic Data

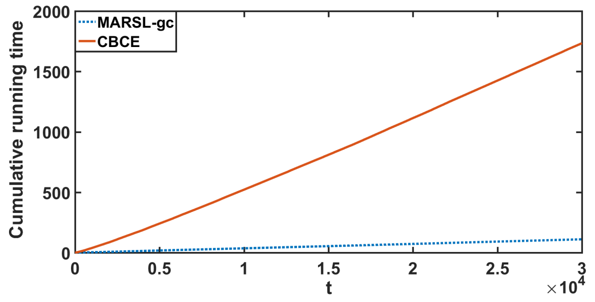
We consider the problem of matrix regression with a nuclear norm regularizer [Bach, 2008]. In each round t, firstly a data point (M_t, y_t) arrives, where $M_t \in \mathbb{R}^{p \times q}$ is a feature matrix and $y_t \in \mathbb{R}$ is the target value. Then, the algorithm predicts a parameter matrix $W_t \in \mathbb{R}^{p \times m}$ without knowing the data point and incurs a loss, defined as:

$$f_t(W_t) = \frac{1}{a} \left[\frac{1}{2} \left(y_t - \mathrm{tr}\left(W_t^\top M_t \right) \right)^2 + b\|W_t\|_* \right], \tag{22}$$

where a and b are constant parameters, and $\|\cdot\|_*$ is the nuclear norm. Inspired by Jun *et al.* [2017], we create a scenario where the optimal parameter matrix is drifting over time. Specifically, we sample M_t, $\forall t = 1, ..., 30000$ and W_i, $\forall i = 1, 2, 3$ uniformly at random from $[-1, 1]^{100 \times 100}$. For $t \in [1, 10000]$, we assign $y_t = \mathrm{tr}(W_1^\top M_t) + \epsilon$, where ϵ is a Gaussian distributed noise. For $t \in [10001, 20000]$ ($[20001, 30000]$), we use the same assignment but with parameter matrix W_2 (W_3). In this way, the optimal parameter matrix will change every 10000 rounds. Since the loss function is convex, CBCE and MARSL-gc can be applied. As

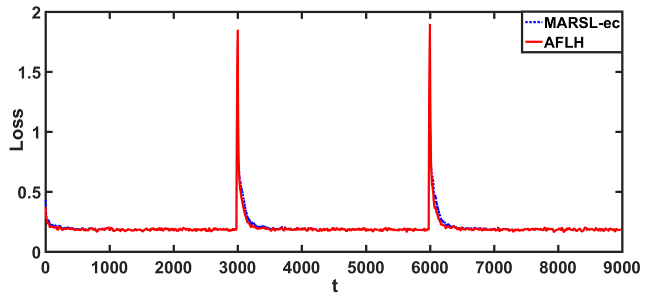


(a) Loss vs. the number of iterations

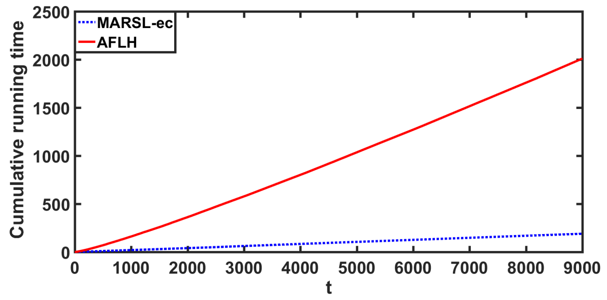


(b) Cumulative running time vs. the number of iterations

Figure 1: Experimental results of nuclear norm regularized matrix regression



(a) Loss vs. the number of iterations



(b) Cumulative running time vs. the number of iterations

Figure 2: Experimental results of mini-batch logistic regression

discussed in Section 3.3, our surrogate loss takes the following form:

$$L_t^{gc}(W) = \partial f_t(W_t)^\top \bullet (W - W_t)/a + 1/2, \tag{23}$$

where $\bullet$ denotes the element-wise product, W_t is the matrix chosen by the meta algorithm, and $\partial f_t(W_t)$ is a subgradient of $f_t(W_t)$, computed by SVD [Watson, 1992].

Following Jun *et al.* [2017], we scale both loss and surrogate loss by assigning $a = 500$ and capping them above at 1. The value of b is empirically set as 10^{-4}. We repeat the experiment 50 times, and the losses of the two algorithms are averaged by a moving time window of length 50, as shown in Figure 1(a). The cumulative running time is reported in Figure 1(b). As can be seen, MARSL-gc performs very closely to CBCE, and MARSL-gc is much more efficient. In fact, it achieves 15.9 times speed-up (1733.9s vs. 110.5s). Although in each round t the number of experts of MARSL-gc and CBCE are the same (i.e., $O(\log t)$), the computations differ dramatically. Namely, in CBCE, *each expert* has to perform SVD twice to compute its loss and gradient, while in MARSL-gc the SVD operations are only performed *in the meta algorithm*. Note that the speed ratio will grow if we increase the number of iterations or the size of the matrix.

4.2 Real-World Data with a Synthetic Component

Next, we consider the problem of mini-batch logistic regression [Shalev-Shwartz and Zhang, 2013]. In each round t, a batch of training examples $\{(\mathbf{x}_{t,1}, y_{t,1}), \ldots, (\mathbf{x}_{t,n}, y_{t,n})\}$ arrives, where $(\mathbf{x}_{t,i}, y_{t,i}) \in [-1,1]^d \times \{-1,1\}, i = 1, \ldots, n$. Simultaneously, the learner makes a prediction of the unknown parameter $\mathbf{w}$, denoted as $\mathbf{w}_t$, and suffers a loss, defined as:

$$f_t(\mathbf{w}_t) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-y_{t,i}\mathbf{w}_t^\top \mathbf{x}_{t,i})). \tag{24}$$

Following the same spirit as in Section 4.1., We build a dynamic scenario based on real-world binary classification data set IJCNN01 [Prokhorov, 2001; Chang and Lin, 2011]. Let the number of iterations be 9000. In the first and last 3000 iterations, the algorithm receives a batch of training data which is randomly sampled from the original data set; in iterations 3001-6000, the labels of samples are flipped by multiplying -1. In this way, the optimal parameter will change every 3000 iterations. To make Assumption 2 is satisfied, we also add a domain constraint such that the optimal parameters are inside a d-dimensional ball with radius 10. Since the loss function is exp-concave, AFLH and MARSL-ec can be applied. As discussed in Section 3.1, the surrogate loss takes the following form:

$$\begin{aligned} L_t^{ec}(\mathbf{u}) =& \frac{\gamma}{2}(\mathbf{w}_t - \mathbf{u})^\top \left(\frac{1}{n}\sum_{i=1}^{n} \nabla_{t,i}\nabla_{t,i}^\top\right)(\mathbf{w}_t - \mathbf{u}) \\ &+ \left(\frac{1}{n}\sum_{i=1}^{n}\nabla_{t,i}\right)^\top (\mathbf{u} - \mathbf{w}_t), \end{aligned} \tag{25}$$

where

$$\nabla_{t,i} = -y_{t,i}\mathbf{x}_{t,i}/\left(1 + \exp(y_{t,i}\mathbf{w}_t^\top \mathbf{x}_{t,i})\right).$$

(25) implies that MARSL-ec can directly compute the averaged gradient in the meta algorithm, and thus it is more efficient. We set $n = 256$ for both algorithms. the experiment is repeated 10 times, and the losses are averaged by a moving time window of length 20, as shown in Figure 2(a). The cumulative running time is reported in Figure 2(b). It can be seen that MARSL-ec performs nearly as well as AFLH, and achieves 10.6 times speed-up (2010.7s vs. 189.5s). The speed ratio will also grow if we increase the number of iterations or the size of the batch.

5 Conclusion

In this paper, we propose a series of efficient algorithms for minimizing the adaptive regret of exp-concave, strongly convex and general convex functions respectively. The main advantage of our algorithms is that they only require one gradient query per round, and attain the same theoretical guarantees as previous optimal algorithms. Empirical results on different data sets demonstrate the efficiency and effectiveness of our methods.

Acknowledgements

This work was partially supported by the National Key R&D Program of China (2018YFB1004300), NSFC (61603177), JiangsuSF (BK20160658), YESS (2017QNRC001), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[Abernethy *et al.*, 2009] Jacob Abernethy, Alekh Agarwal, and Peter L Bartlett. A stochastic view of optimal regret through minimax duality. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

[Bach, 2008] Francis R Bach. Consistency of trace norm minimization. *Journal of Machine Learning Research*, 9:1019–1048, 2008.

[Blum, 1997] Avrim Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.

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

[Cesa-Bianchi and Lugosi, 2006] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011.

[Daniely *et al.*, 2015] Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1405–1411, 2015.

[Hazan and Seshadhri, 2007] Elad Hazan and C Seshadhri. Adaptive algorithms for online decision problems. In *Electronic Colloquium on Computational Complexity*, 2007.

[Hazan and Seshadhri, 2009] Elad Hazan and Comandur Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th International Conference on Machine Learning*, pages 393–400, 2009.

[Hazan *et al.*, 2007] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169–192, 2007.

[Hazan, 2016] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2:157–325, 2016.

[Ji and Ye, 2009] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th International Conference on Machine Learning*, pages 457–464, 2009.

[Jun *et al.*, 2017] Kwang-Sung Jun, Francesco Orabona, Stephen Wright, and Rebecca Willett. Improved strongly adaptive online learning using coin betting. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 943–951, 2017.

[Li *et al.*, 2014] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 661–670, 2014.

[Orabona and Pál, 2016] Francesco Orabona and Dávid Pál. Coin betting and parameter-free online learning. In *Advances in Neural Information Processing Systems 29*, pages 577–585, 2016.

[Prokhorov, 2001] Danil Prokhorov. Ijcnn 2001 neural network competition. *Slide Presentation in IJCNN*, 1:97, 2001.

[Shalev-Shwartz and Zhang, 2013] Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems 26*, pages 378–385, 2013.

[Shalev-Shwartz, 2012] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4:107–194, 2012.

[van Erven and Koolen, 2016] Tim van Erven and Wouter M Koolen. Metagrad: Multiple learning rates in online learning. In *Advances in Neural Information Processing Systems 29*, pages 3666–3674, 2016.

[Watson, 1992] G Alistair Watson. Characterization of the subdifferential of some matrix norms. *Linear Algebra and Its Applications*, 170:33–45, 1992.

[Zhang *et al.*, 2017] Lijun Zhang, Tianbao Yang, Rong Jin, and Zhi-Hua Zhou. Dynamic regret of strongly adaptive methods. *arXiv preprint arXiv:1701.07570*, 2017.

[Zinkevich, 2003] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.