

Optimal Black-Box Reductions Between Optimization Objectives

Zeyuan Allen-Zhu
zeyuan@csail.mit.edu
Princeton University

Elad Hazan
ehazan@cs.princeton.edu
Princeton University

first circulated on February 5, 2016*

Abstract

The diverse world of machine learning applications has given rise to a plethora of algorithms and optimization methods, finely tuned to the specific regression or classification task at hand. We reduce the complexity of algorithm design for machine learning by reductions: we develop reductions that take a method developed for one setting and apply it to the entire spectrum of smoothness and strong-convexity in applications.

Furthermore, unlike existing results, our new reductions are *optimal* and more *practical*. We show how these new reductions give rise to new and faster running times on training linear classifiers for various families of loss functions, and conclude with experiments showing their successes also in practice.

1 Introduction

The basic machine learning problem of minimizing a regularizer plus a loss function comes in numerous different variations and names. Examples include Ridge Regression, Lasso, Support Vector Machine (SVM), Logistic Regression and many others. A multitude of optimization methods were introduced for these problems, but in most cases specialized to very particular problem settings. Such specializations appear necessary since objective functions for different classification and regularization tasks admit different convexity and smoothness parameters. We list below a few recent algorithms along with their applicable settings.

- Variance-reduction methods such as SAGA and SVRG [7, 12] *intrinsically require* the objective to be *smooth*, and do not work for non-smooth problems like SVM. This is because for loss functions such as hinge loss, no unbiased gradient estimator can achieve a variance that approaches to zero.
- Dual methods such as SDCA or APCG [18, 28] *intrinsically require* the objective to be *strongly convex (SC)*, and do not directly apply to non-SC problems. This is because for a non-SC objective such as Lasso, its dual is not even well-defined.
- Primal-dual methods such as SPDC [32] require the objective to be both smooth and SC. Many other algorithms are only analyzed for both smooth and SC objectives [5, 14, 15].

In this paper we investigate whether such specializations are inherent. Is it possible to take a convex optimization algorithm designed for one problem, and apply it to different classification or

*First appeared on ArXiv on March 17, 2016. Corrected a few typos in this most recent version.

regression settings in a black-box manner? Such a reduction should ideally take full and *optimal* advantage of the objective properties, namely strong-convexity and smoothness, for each setting.

Unfortunately, existing reductions are still very limited for at least two reasons. First, they incur at least a logarithmic factor $\log(1/\varepsilon)$ in the running time so leading only to suboptimal convergence rates.¹ Second, after applying existing reductions, algorithms become *biased* so the objective value does not converge to the global minimum. These theoretical concerns also translate into running time losses and parameter tuning difficulties in practice.

In this paper, we develop new and *optimal* regularization and smoothing reductions that can

- shave off a non-optimal $\log(1/\varepsilon)$ factor
- produce unbiased algorithms

Besides such technical advantages, our new reductions also enable researchers to focus on designing algorithms for only one setting but infer optimal results more broadly. This is opposed to results such as [4, 23] where the authors develop *ad hoc* techniques to tweak specific algorithms, rather than all algorithms, and apply them to other settings without losing extra factors and without introducing bias.

Our new reductions also enable researchers to prove *lower bounds* more broadly [30].

1.1 Formal Setting and Classical Approaches

Consider minimizing a composite objective function

$$\min_{x \in \mathbb{R}^d} \{F(x) \stackrel{\text{def}}{=} f(x) + \psi(x)\} , \quad (1.1)$$

where $f(x)$ is a differentiable convex function and $\psi(x)$ is a relatively simple (but possibly non-differentiable) convex function, sometimes referred to as the *proximal* function. Our goal is to find a point $x \in \mathbb{R}^d$ satisfying $F(x) \leq F(x^*) + \varepsilon$, where x^* is a minimizer of F .

In most classification and regression problems, $f(x)$ can be written as $f(x) = \frac{1}{n} \sum_{i=1}^n f_i(\langle x, a_i \rangle)$ where each $a_i \in \mathbb{R}^d$ is a feature vector. We refer to this as the *finite-sum* case of (1.1).

- CLASSICAL REGULARIZATION REDUCTION.

Given a non-SC $F(x)$, one can define a new objective $F'(x) \stackrel{\text{def}}{=} F(x) + \frac{\sigma}{2} \|x_0 - x\|^2$ in which σ is on the order of ε . In order to minimize $F(x)$, the classical regularization reduction calls an oracle algorithm to minimize $F'(x)$ instead, and this oracle only needs to work with SC functions.

EXAMPLE. If F is L -smooth, one can apply accelerated gradient descent to minimize F' and obtain an algorithm that converges in $O(\sqrt{L}/\varepsilon \log \frac{1}{\varepsilon})$ iterations in terms of minimizing the original F . This complexity has a suboptimal dependence on ε and shall be improved using our new regularization reduction.

- CLASSICAL SMOOTHING REDUCTION (FINITE-SUM CASE).

Given a non-smooth $F(x)$ of a finite-sum form,² one can define a smoothed variant $\widehat{f}_i(\alpha) =$

¹Recall that obtaining the *optimal* convergence rate is one of the main goals in operations research and machine learning. For instance, obtaining the optimal $1/\varepsilon$ rate for online learning was a major breakthrough since the $\log(1/\varepsilon)/\varepsilon$ rate was discovered [11, 13, 24].

²Smoothing reduction is typically applied to the finite sum form only. This is because, for a general high dimensional function $f(x)$, its smoothed variant $\widehat{f}(x)$ may not be efficiently computable.

$\mathbb{E}_{v \in [-1,1]}[f_i(\alpha + \varepsilon v)]$ for each $f_i(\alpha)$ and let $F'(x) = \frac{1}{n} \sum_{i=1}^n \widehat{f}_i(\langle a_i, x \rangle) + \psi(x)$.³ In order to minimize $F(x)$, the classical smoothing reduction calls an oracle algorithm to minimize $F'(x)$ instead, and this oracle only needs to work with smooth functions.

EXAMPLE. If $F(x)$ is σ -SC and one applies accelerated gradient descent to minimize F' , this yields an algorithm that converges in $O(\frac{1}{\sqrt{\sigma\varepsilon}} \log \frac{1}{\varepsilon})$ iterations for minimizing the original $F(x)$. Again, the additional factor $\log(1/\varepsilon)$ can be removed using our new smoothing reduction.

Besides the non-optimality, applying the above two reductions gives only *biased* algorithms. One has to tune the regularization or smoothing parameter, and the algorithm only converges to the minimum of the regularized or smoothed problem $F'(x)$, which can be away from the true minimizer of $F(x)$ by a distance proportional to the parameter. This makes the reduction hard to use in practice.

1.2 Our New Results

To introduce our new reductions, we first define a property on the oracle algorithm.

Our Black-Box Oracle. Consider an algorithm \mathcal{A} that minimizes (1.1) when the objective F is L -smooth and σ -SC. We say that \mathcal{A} satisfies the *homogenous objective decrease* (HOOD) property in time $\text{Time}(L, \sigma)$ if, for every starting vector x_0 , \mathcal{A} produces an output x' satisfying $F(x') - F(x^*) \leq \frac{F(x_0) - F(x^*)}{4}$ in time $\text{Time}(L, \sigma)$. In other words, \mathcal{A} decreases the objective value distance to the minimum by a constant factor in time $\text{Time}(L, \sigma)$, regardless of how large or small $F(x_0) - F(x^*)$ is. We give a few example algorithms that satisfy HOOD:

- Gradient descent and accelerated gradient descent satisfy HOOD with $\text{Time}(L, \sigma) = O(L/\sigma) \cdot C$ and $\text{Time}(L, \sigma) = O(\sqrt{L/\sigma}) \cdot C$ respectively, where C is the time needed to compute a gradient $\nabla f(x)$ and perform a proximal gradient update [21]. Many subsequent works in this line of research also satisfy HOOD, including [3, 5, 14, 15].
- SVRG and SAGA [12, 31] solve the finite-sum form of (1.1) and satisfy HOOD with $\text{Time}(L, \sigma) = O(n + L/\sigma) \cdot C_1$ where C_1 is the time needed to compute a stochastic gradient $\nabla f_i(x)$ and perform a proximal gradient update.
- Katyusha [1] solves the finite-sum form of (1.1) and satisfies HOOD with $\text{Time}(L, \sigma) = O(n + \sqrt{nL/\sigma}) \cdot C_1$.

AdaptReg. For objectives $F(x)$ that are non-SC and L -smooth, our **AdaptReg** reduction calls the an oracle satisfying HOOD a logarithmic number of times, each time with a SC objective $F(x) + \frac{\sigma}{2} \|x - x_0\|^2$ for an exponentially decreasing value σ . In the end, **AdaptReg** produces an output \widehat{x} satisfying $F(\widehat{x}) - F(x^*) \leq \varepsilon$ with a total running time $\sum_{t=0}^{\infty} \text{Time}(L, \varepsilon \cdot 2^t)$.

Since most algorithms have an inverse polynomial dependence on σ in $\text{Time}(L, \sigma)$, when summing up $\text{Time}(L, \varepsilon \cdot 2^t)$ for positive values t , we do not incur the additional factor $\log(1/\varepsilon)$ as opposed to the old reduction. In addition, **AdaptReg** is an *unbiased* and *anytime* algorithm. $F(\widehat{x})$ converges to $F(x^*)$ as the time goes without the necessity of changing parameters, so the algorithm can be interrupted at any time. We mention some theoretical applications of **AdaptReg**:

³More formally, one needs this variant to satisfy $|\widehat{f}_i(\alpha) - f_i(\alpha)| \leq \varepsilon$ for all α and be smooth at the same time. This can be done at least in two classical ways if $\widehat{f}_i(\alpha)$ is Lipschitz continuous. One is to define $\widehat{f}_i(\alpha) = \mathbb{E}_{v \in [-1,1]}[f_i(\alpha + \varepsilon v)]$ as an integral of f over the scaled unit interval, see for instance Chapter 2.3 of [10], and the other is to define $\widehat{f}_i(\alpha) = \max_{\beta} \{\beta \cdot \alpha - f_i^*(\beta) - \frac{\varepsilon}{2} \alpha^2\}$ using the Fenchel dual $f_i^*(\beta)$ of $f_i(\alpha)$, see for instance [22].

- Applying **AdaptReg** to SVRG, we obtain a running time $O(n \log \frac{1}{\varepsilon} + \frac{L}{\varepsilon}) \cdot C_1$ for minimizing finite-sum, non-SC, and smooth objectives (such as Lasso and Logistic Regression). This improves on known theoretical running time obtained by non-accelerated methods, including $O(n \log \frac{1}{\varepsilon} + \frac{L}{\varepsilon} \log \frac{1}{\varepsilon}) \cdot C_1$ through the old reduction, as well as $O(\frac{n+L}{\varepsilon}) \cdot C_1$ through direct methods such as SAGA [7] and SAG [25].
- Applying **AdaptReg** to Katyusha, we obtain a running time $O(n \log \frac{1}{\varepsilon} + \frac{\sqrt{nL}}{\sqrt{\varepsilon}}) \cdot C_1$ for minimizing finite-sum, non-SC, and smooth objectives (such as Lasso and Logistic Regression). This is the first and only known stochastic method that converges with the optimal $1/\sqrt{\varepsilon}$ rate (as opposed to $\log(1/\varepsilon)/\sqrt{\varepsilon}$) for this class of objectives. [1]
- Applying **AdaptReg** to methods that do not originally work for non-SC objectives such as [5, 14, 15], we improve their running times by a factor of $\log(1/\varepsilon)$ for working with non-SC objectives.

AdaptSmooth and JointAdaptRegSmooth. For objectives $F(x)$ that are finite-sum, σ -SC, but non-smooth, our **AdaptSmooth** reduction calls an oracle satisfying **HOOD** a logarithmic number of times, each time with a smoothed variant of $F^{(\lambda)}(x)$ and an exponentially decreasing smoothing parameter λ . In the end, **AdaptSmooth** produces an output \hat{x} satisfying $F(\hat{x}) - F(x^*) \leq \varepsilon$ with a total running time $\sum_{t=0}^{\infty} \text{Time}(\frac{1}{\varepsilon \cdot 2^t}, \sigma)$.

Since most algorithms have a polynomial dependence on L in $\text{Time}(L, \sigma)$, when summing up $\text{Time}(\frac{1}{\varepsilon \cdot 2^t}, \sigma)$ for positive values t , we do not incur an additional factor of $\log(1/\varepsilon)$ as opposed to the old reduction. **AdaptSmooth** is also an *unbiased* and *anytime* algorithm for the same reason as **AdaptReg**.

In addition, **AdaptReg** and **AdaptSmooth** can effectively work together, to solve finite-sum, non-SC, and non-smooth case of (1.1), and we call this reduction **JointAdaptRegSmooth**.

We mention some theoretical applications of **AdaptSmooth** and **JointAdaptRegSmooth**:

- Applying **AdaptReg** to Katyusha, we obtain a running time $O(n \log \frac{1}{\varepsilon} + \frac{\sqrt{n}}{\sqrt{\sigma\varepsilon}}) \cdot C_1$ for minimizing finite-sum, SC, and non-smooth objectives (such as SVM). Therefore, Katyusha combined with **AdaptReg** is the first and only known stochastic method that converges with the optimal $1/\sqrt{\varepsilon}$ rate (as opposed to $\log(1/\varepsilon)/\sqrt{\varepsilon}$) for this class of objectives. [1]
- Applying **JointAdaptRegSmooth** to Katyusha, we obtain a running time $O(n \log \frac{1}{\varepsilon} + \frac{\sqrt{n}}{\varepsilon}) \cdot C_1$ for minimizing finite-sum, SC, and non-smooth objectives (such as L1-SVM). Therefore, Katyusha combined with **JointAdaptRegSmooth** is the first and only known stochastic method that converges with the optimal $1/\varepsilon$ rate (as opposed to $\log(1/\varepsilon)/\varepsilon$) for this class of objectives. [1]

Theory vs. Practice. In theory, not all algorithms solving (1.1) satisfy **HOOD**. Some machine learning algorithms such as APCG [18], SPDC [32], AccSDCA [29] and SDCA [28] either do not satisfy **HOOD** or incur some additional $\log(L/\sigma)$ factor in its running time so cannot benefit from our new reductions in theory. For example, APCG solves the finite-sum form of (1.1) and produces an output x satisfying $F(x) - F(x^*) \leq \varepsilon$ in time $O((n + \frac{\sqrt{nL}}{\sqrt{\sigma}}) \cdot \log(\frac{L}{\sigma\varepsilon})) \cdot C_1$. This running time does not have a logarithmic dependence on ε that has the form $\log(\frac{F(x_0) - F(x^*)}{\varepsilon})$. In other words, APCG might in principle take a much longer running time in order to decrease the objective distance to the minimum from 1 to $1/4$, as compared to the time needed to decrease from 10^{-10} to $10^{-10}/4$.

Fortunately, although without theoretical guarantee, these methods also benefit from our new reductions, and we include experiments in this paper to confirm such findings.

Related Works. Catalyst and APPA [9, 17] reductions turn non-accelerated methods into accelerated ones. They can be used as regularization reductions too; however, in such a case they become identical to the traditional regularization reduction, and continue to introduce bias and suffer from a log factor loss in the running time. In fact, Catalyst and APPA fix the regularization parameter throughout the algorithm but our **AdaptReg** decreases it exponentially. Therefore, their results cannot imply ours.

PRISMA [23] turns Nesterov’s accelerated gradient descent to work for non-smooth objectives without paying the log factor. However, PRISMA does not apply to all algorithms in a black-box manner so is not a reduction. Furthermore, PRISMA requires the algorithm to know the number of iterations in advance, which **AdaptSmooth** does not .

Roadmap. We include the description and analysis of **AdaptReg** in Section 3, but only include the description of **AdaptSmooth** in Section 4. We leave proofs as well as the description and analysis of **JointAdaptRegSmooth** to the appendix. We include experimental results in Section 6.

2 Preliminaries

In this paper we denote by $\nabla f(x)$ the full gradient of f if it is differentiable, or the subgradient if f is only Lipschitz continuous. Recall some classical definitions on strong convexity and smoothness.

Definition 2.1 (smoothness and strong convexity). *For a convex function $f: \mathbb{R}^n \rightarrow \mathbb{R}$,*

- *f is σ -strongly convex if $\forall x, y \in \mathbb{R}^n$, it satisfies $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\sigma}{2} \|x - y\|^2$.*
- *f is L -smooth if $\forall x, y \in \mathbb{R}^n$, it satisfies $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$.*

Characterization of SC and Smooth Regimes. In this paper we give numbers to the following 4 categories of objectives $F(x)$ in (1.1). Each of them corresponds to some well-known training problems in machine learning. (Letting $(a_i, b_i) \in \mathbb{R}^d \times \mathbb{R}$ be the i -th feature vector and label.)

Case 1: $\psi(x)$ is σ -SC and $f(x)$ is L -smooth. Examples:

- *ridge regression: $f(x) = \frac{1}{2n} \sum_{i=1}^n (\langle a_i, x \rangle - b_i)^2$ and $\psi(x) = \frac{\sigma}{2} \|x\|_2^2$.*
- *elastic net: $f(x) = \frac{1}{2n} \sum_{i=1}^n (\langle a_i, x \rangle - b_i)^2$ and $\psi(x) = \frac{\sigma}{2} \|x\|_2^2 + \lambda \|x\|_1$.*

Case 2: $\psi(x)$ is non-SC and $f(x)$ is L -smooth. Examples:

- *Lasso: $f(x) = \frac{1}{2n} \sum_{i=1}^n (\langle a_i, x \rangle - b_i)^2$ and $\psi(x) = \lambda \|x\|_1$.*
- *logistic regression: $f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i \langle a_i, x \rangle))$ and $\psi(x) = \lambda \|x\|_1$.*

Case 3: $\psi(x)$ is σ -SC and $f(x)$ is non-smooth (but Lipschitz continuous). Examples:

- *SVM: $f(x) = \frac{1}{n} \sum_{i=1}^n \max\{0, 1 - b_i \langle a_i, x \rangle\}$ and $\psi(x) = \sigma \|x\|_2^2$.*

Case 4: $\psi(x)$ is non-SC and $f(x)$ is non-smooth (but Lipschitz continuous). Examples:

- *ℓ_1 -SVM: $f(x) = \frac{1}{n} \sum_{i=1}^n \max\{0, 1 - b_i \langle a_i, x \rangle\}$ and $\psi(x) = \lambda \|x\|_1$.*

Definition 2.2 (HOOD property). *We say an algorithm $\mathcal{A}(F, x_0)$ solving Case 1 of problem (1.1) satisfies the homogenous objective decrease (HOOD) property with time $\text{Time}(L, \sigma)$ if, for every starting point x_0 , it produces output $x' \leftarrow \mathcal{A}(F, x_0)$ such that $F(x') - \min_x F(x) \leq \frac{F(x_0) - \min_x F(x)}{4}$ in time $\text{Time}(L, \sigma)$.⁴*

⁴Although our definition is only for deterministic algorithms, if the guarantee is probabilistic, i.e., $\mathbb{E}[F(x')] - \min_x F(x) \leq \frac{F(x_0) - \min_x F(x)}{4}$, all the results of this paper remain true.

Algorithm 1 The AdaptReg Reduction

Input: an objective $F(\cdot)$ in Case 2 (smooth and not necessarily strongly convex);
 x_0 a starting vector, σ_0 an initial regularization parameter, T the number of epochs;
an algorithm \mathcal{A} that solves Case 1 of problem (1.1).

Output: \hat{x}_T .

- 1: $\hat{x}_0 \leftarrow x_0$.
 - 2: **for** $t \leftarrow 0$ **to** $T - 1$ **do**
 - 3: Define $F^{(\sigma_t)}(x) \stackrel{\text{def}}{=} \frac{\sigma_t}{2} \|x - x_0\| + F(x)$.
 - 4: $\hat{x}_{t+1} \leftarrow \mathcal{A}(F^{(\sigma_t)}, \hat{x}_t)$.
 - 5: $\sigma_{t+1} \leftarrow \sigma_t/2$.
 - 6: **end for**
 - 7: **return** \hat{x}_T .
-

In this paper, we denote by C the time needed for computing a full gradient $\nabla f(x)$ and performing a proximal gradient update of the form $x' \leftarrow \arg \min_x \{ \frac{1}{2} \|x - x_0\|^2 + \eta(\langle \nabla f(x), x - x_0 \rangle + \psi(x)) \}$. For the finite-sum case of problem (1.1), we denote by C_1 the time needed for computing a stochastic (sub-)gradient $\nabla f_i(\langle a_i, x \rangle)$ and performing a proximal gradient update of the form $x' \leftarrow \arg \min_x \{ \frac{1}{2} \|x - x_0\|^2 + \eta(\langle \nabla f_i(\langle a_i, x \rangle) a_i, x - x_0 \rangle + \psi(x)) \}$. For finite-sum forms of (1.1), C is usually on the magnitude of $n \times C_1$.

3 AdaptReg: Reduction from Case 2 to Case 1

We now focus on solving Case 2 of problem (1.1): that is, $f(\cdot)$ is L -smooth, but $\psi(\cdot)$ is not necessarily SC. We achieve so by reducing the problem to an algorithm \mathcal{A} solving Case 1 that satisfies HOOD.

AdaptReg works as follows (see Algorithm 1). At the beginning of AdaptReg, we set \hat{x}_0 to equal x_0 , an arbitrary given starting vector. AdaptReg consists of T epochs. At each epoch $t = 0, 1, \dots, T - 1$, we define a σ_t -strongly convex objective $F^{(\sigma_t)}(x) \stackrel{\text{def}}{=} \frac{\sigma_t}{2} \|x - x_0\|^2 + F(x)$. Here, the parameter $\sigma_{t+1} = \sigma_t/2$ for each $t \geq 0$ and σ_0 is an input parameter to AdaptReg that will be specified later. We run \mathcal{A} on $F^{(\sigma_t)}(x)$ with starting vector \hat{x}_t in each epoch, and let the output be \hat{x}_{t+1} . After all T epochs are finished, AdaptReg simply outputs \hat{x}_T .

We state our main theorem for AdaptReg below and prove it in Section 3.1.

Theorem 3.1 (AdaptReg). *Suppose that in problem (1.1) $f(\cdot)$ is L -smooth. Let x_0 be a starting vector such that $F(x_0) - F(x^*) \leq \Delta$ and $\|x_0 - x^*\|^2 \leq \Theta$. Then, AdaptReg with $\sigma_0 = \Delta/\Theta$ and $T = \log_2(\Delta/\varepsilon)$ produces an output \hat{x}_T satisfying $F(\hat{x}_T) - \min_x F(x) \leq O(\varepsilon)$ in a total running time of $\sum_{t=0}^{T-1} \text{Time}(L, \sigma_0 \cdot 2^{-t})$.⁵*

Remark 3.2. We compare the parameter tuning effort needed for AdaptReg against the classical regularization reduction. In the classical reduction, there are two parameters: T , the number of iterations that does not need tuning; and σ , which had better equal ε/Θ which is an unknown quantity so requires tuning. In AdaptReg, we also need tune only one parameter, that is σ_0 . Our T need not be tuned because AdaptReg can be interrupted at any moment and \hat{x}_t of the current epoch can be outputted. In our experiments later, we spent the *same effort* turning σ in the classical reduction and σ_0 in AdaptReg. As it can be easily seen from the plots, tuning σ_0 is much easier than σ .

⁵If the HOOD property is only satisfied probabilistically as per Footnote 4, our error guarantee becomes probabilistic, i.e., $\mathbb{E}[F(\hat{x}_T)] - \min_x F(x) \leq O(\varepsilon)$. This is also true for other reduction theorems of this paper.

Corollary 3.3. *When AdaptReg is applied to SVRG, we solve the finite-sum case of Case 2 with running time $\sum_{t=0}^{T-1} \text{Time}(L, \sigma_0 \cdot 2^{-t}) = \sum_{t=0}^{T-1} O(n + \frac{L2^t}{\sigma_0}) \cdot C_1 = O(n \log \frac{\Delta}{\varepsilon} + \frac{L\Theta}{\varepsilon}) \cdot C_1$. This is faster than $O((n + \frac{L\Theta}{\varepsilon}) \log \frac{\Delta}{\varepsilon}) \cdot C_1$ obtained through the old reduction, and faster than $O(\frac{n+L\Theta}{\varepsilon}) \cdot C_1$ obtained by SAGA [7] and SAG [25].*

When AdaptReg is applied to Katyusha, we solve the finite-sum case of Case 2 with running time $\sum_{t=0}^{T-1} \text{Time}(L, \sigma_0 \cdot 2^{-t}) = \sum_{t=0}^{T-1} O(n + \frac{\sqrt{nL}2^t}{\sqrt{\sigma_0}}) \cdot C_1 = O(n \log \frac{\Delta}{\varepsilon} + \sqrt{nL\Theta/\varepsilon}) \cdot C_1$. This is faster than $O((n + \sqrt{nL/\varepsilon}) \log \frac{\Delta}{\varepsilon}) \cdot C_1$ obtained through the old reduction on Katyusha [1].⁶

3.1 Convergence Analysis for AdaptReg

For analysis purpose, we define x_{t+1} to be the exact minimizer of $F^{(\sigma_t)}(x)$. The HOOD property of \mathcal{A} ensures that

$$F^{(\sigma_t)}(\hat{x}_{t+1}) - F^{(\sigma_t)}(x_{t+1}) \leq \frac{F^{(\sigma_t)}(\hat{x}_t) - F^{(\sigma_t)}(x_{t+1})}{4}. \quad (3.1)$$

We denote by x^* an arbitrary minimizer of $F(x)$, and the following claim states a simple property about the minimizers of $F^{(\sigma_t)}(x)$:

Claim 3.4. *We have $\|x_{t+1} - x^*\| \leq \|x_0 - x^*\|$ for each $t \geq 0$.*

Proof. By the strong convexity of $F^{(\sigma_t)}(x)$ and the fact that x_{t+1} is its exact minimizer, we have

$$F^{(\sigma_t)}(x_{t+1}) - F^{(\sigma_t)}(x^*) \leq -\frac{\sigma_t}{2} \|x_{t+1} - x^*\|^2.$$

Using the fact that $F^{(\sigma_t)}(x_{t+1}) \geq F(x_{t+1})$, as well as the definition $F^{(\sigma_t)}(x^*) = \frac{\sigma_t}{2} \|x^* - x_0\|^2 + F(x^*)$, we immediately have

$$\frac{\sigma_t}{2} \|x_0 - x^*\|^2 - \frac{\sigma_t}{2} \|x_{t+1} - x^*\|^2 \geq F(x_{t+1}) - F(x^*) \geq 0. \quad \square$$

Define $D_t \stackrel{\text{def}}{=} F^{(\sigma_t)}(\hat{x}_t) - F^{(\sigma_t)}(x_{t+1})$ to be the initial objective distance to the minimum on function $F^{(\sigma_t)}$ before we call \mathcal{A} in epoch t . At epoch 0, we have upper bound $D_0 = F^{(\sigma_0)}(\hat{x}_0) - \min_x F^{(\sigma_0)}(x) \leq F(x_0) - F(x^*)$. For each epoch $t \geq 1$, we compute that

$$\begin{aligned} D_t &\stackrel{\text{def}}{=} F^{(\sigma_t)}(\hat{x}_t) - F^{(\sigma_t)}(x_{t+1}) \\ &\stackrel{\text{①}}{=} F^{(\sigma_{t-1})}(\hat{x}_t) - \frac{\sigma_{t-1} - \sigma_t}{2} \|x_0 - \hat{x}_t\|^2 - F^{(\sigma_{t-1})}(x_{t+1}) + \frac{\sigma_{t-1} - \sigma_t}{2} \|x_0 - x_{t+1}\|^2 \\ &\stackrel{\text{②}}{\leq} F^{(\sigma_{t-1})}(\hat{x}_t) - \frac{\sigma_{t-1} - \sigma_t}{2} \|x_0 - \hat{x}_t\|^2 - F^{(\sigma_{t-1})}(x_t) - \frac{\sigma_{t-1}}{2} \|x_t - x_{t+1}\|^2 + \frac{\sigma_{t-1} - \sigma_t}{2} \|x_0 - x_{t+1}\|^2 \\ &\leq F^{(\sigma_{t-1})}(\hat{x}_t) - F^{(\sigma_{t-1})}(x_t) + \frac{\sigma_{t-1} - \sigma_t}{2} \|x_0 - x_{t+1}\|^2 \\ &\stackrel{\text{③}}{\leq} F^{(\sigma_{t-1})}(\hat{x}_t) - F^{(\sigma_{t-1})}(x_t) + \frac{\sigma_{t-1} - \sigma_t}{2} (2\|x_0 - x^*\|^2 + 2\|x_{t+1} - x^*\|^2) \\ &\stackrel{\text{④}}{\leq} F^{(\sigma_{t-1})}(\hat{x}_t) - F^{(\sigma_{t-1})}(x_t) + 2(\sigma_{t-1} - \sigma_t) \|x_0 - x^*\|^2 \\ &\stackrel{\text{⑤}}{\leq} \frac{D_{t-1}}{4} + 2(\sigma_{t-1} - \sigma_t) \|x_0 - x^*\|^2 \stackrel{\text{⑥}}{=} \frac{D_{t-1}}{4} + 2\sigma_t \|x_0 - x^*\|^2. \end{aligned}$$

Above, ① follows from the definition of $F^{(\sigma_t)}(\cdot)$ and $F^{(\sigma_{t-1})}(\cdot)$; ② follows from the strong convexity of $F^{(\sigma_{t-1})}(\cdot)$ as well as the fact that x_t is its minimizer; ③ follows because for any two vectors a, b

⁶If the old reduction is applied on APCG, SPDC, or AccSDCA rather than Katyusha, then two log factors will be lost.

it satisfies $\|a - b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$; ④ follows from Claim 3.4; ⑤ follows from the definition of \mathbf{D}_{t-1} and (3.1); and ⑥ uses the choice that $\sigma_t = \sigma_{t-1}/2$ for $t \geq 1$. Recursively applying the above inequality, we have

$$\mathbf{D}_T \leq \frac{\mathbf{D}_0}{4^T} + \|x_0 - x^*\|^2 \cdot \left(2\sigma_T + \frac{2\sigma_{T-1}}{4} + \dots\right) \leq \frac{1}{4^T}(F(x_0) - F(x^*)) + 4\sigma_T\|x_0 - x^*\|^2, \quad (3.2)$$

where the second inequality uses our choice $\sigma_t = \sigma_{t-1}/2$. In sum, we obtain a vector \hat{x}_T satisfying

$$\begin{aligned} F(\hat{x}_T) - F(x^*) &\stackrel{\textcircled{1}}{\leq} F^{(\sigma_T)}(\hat{x}_T) - F^{(\sigma_T)}(x^*) + \frac{\sigma_T}{2}\|x_0 - x^*\|^2 \stackrel{\textcircled{2}}{\leq} F^{(\sigma_T)}(\hat{x}_T) - F^{(\sigma_T)}(x_{T+1}) \\ &\quad + \frac{\sigma_T}{2}\|x_0 - x^*\|^2 \stackrel{\textcircled{3}}{=} \mathbf{D}_T + \frac{\sigma_T}{2}\|x_0 - x^*\|^2 \stackrel{\textcircled{4}}{\leq} \frac{1}{4^T}(F(x_0) - F(x^*)) + 4.5\sigma_T\|x_0 - x^*\|^2. \end{aligned} \quad (3.3)$$

Above, ① uses the fact that $F^{(\sigma_T)}(x) \geq F(x)$ for every x ; ② uses the definition that x_{T+1} is the minimizer of $F^{(\sigma_T)}(\cdot)$; ③ uses the definition of \mathbf{D}_T ; and ④ uses (3.2).

Finally, after appropriately choosing σ_0 and T , (3.3) directly implies Theorem 3.1.

4 AdaptSmooth: Reduction from Case 3 to 1

We now focus on solving the finite-sum form of Case 3 for problem (1.1). That is,

$$\min_x F(x) = \frac{1}{n} \sum_{i=1}^n f_i(\langle a_i, x \rangle) + \psi(x),$$

where $\psi(x)$ is σ -strongly convex and each $f_i(\cdot)$ may not be smooth (but is Lipschitz continuous). Without loss of generality, we assume $\|a_i\| = 1$ for each $i \in [n]$ because otherwise one can scale f_i accordingly. We solve this problem by reducing it to an oracle \mathcal{A} which solves the finite-sum form of Case 1 and satisfies HOOD.

Recall the following definition using Fenchel conjugate:⁷

Definition 4.1. For each function $f_i: \mathbb{R} \rightarrow \mathbb{R}$, let $f_i^*(\beta) \stackrel{\text{def}}{=} \max_{\alpha} \{\alpha \cdot \beta - f_i(\alpha)\}$ be its Fenchel conjugate. Then, we define the following smoothed variant of f_i parameterized by $\lambda > 0$:

$$f_i^{(\lambda)}(\alpha) \stackrel{\text{def}}{=} \max_{\beta} \left\{ \beta \cdot \alpha - f_i^*(\beta) - \frac{\lambda}{2}\beta^2 \right\}.$$

Accordingly, we define

$$F^{(\lambda)}(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i^{(\lambda)}(\langle a_i, x \rangle) + \psi(x).$$

From the property of Fenchel conjugate (see for instance the textbook [26]), we know that $f_i^{(\lambda)}(\cdot)$ is a $(1/\lambda)$ -smooth function and therefore the objective $F^{(\lambda)}(x)$ falls into the finite-sum form of Case 1 for problem (1.1) with smoothness parameter $L = 1/\lambda$.

Our AdaptSmooth works as follows (see Algorithm 2 in Appendix B). At the beginning of AdaptSmooth, we set \hat{x}_0 to equal x_0 , an arbitrary given starting vector. AdaptSmooth consists of

⁷For every explicitly given $f_i(\cdot)$, this Fenchel conjugate can be symbolically computed and fed into the algorithm. This pre-process is needed for nearly all known algorithms in order for them to apply to non-smooth settings (such as SVRG, SAGA, SPDC, APCG, SDCA, etc). SGD and its strongly convex variant PEGASOS are the only known methods which do not need this computation. However, they are not accelerated methods.

T epochs. At each epoch $t = 0, 1, \dots, T - 1$, we define a $(1/\lambda_t)$ -smooth objective $F^{(\lambda_t)}(x)$ using Definition 4.1. Here, the parameter $\lambda_{t+1} = \lambda_t/2$ for each $t \geq 0$ and λ_0 is an input parameter to `AdaptSmooth` that will be specified later. We run \mathcal{A} on $F^{(\lambda_t)}(x)$ with starting vector \hat{x}_t in each epoch, and let the output be \hat{x}_{t+1} . After all T epochs are finished, `AdaptSmooth` outputs \hat{x}_T . (Alternatively, if one sets T to be infinity, `AdaptSmooth` can be interrupted at an arbitrary moment and output \hat{x}_t of the current epoch.)

We state our main theorem for `AdaptSmooth` below and prove it in Appendix B.

Theorem 4.2. *Suppose that in problem (1.1), $\psi(\cdot)$ is σ strongly convex and each $f_i(\cdot)$ is G -Lipschitz continuous. Let x_0 be a starting vector such that $F(x_0) - F(x^*) \leq \Delta$. Then, `AdaptSmooth` with $\lambda_0 = \Delta/G^2$ and $T = \log_2(\Delta/\varepsilon)$ produces an output \hat{x}_T satisfying $F(\hat{x}_T) - \min_x F(x) \leq O(\varepsilon)$ in a total running time of $\sum_{t=0}^{T-1} \text{Time}(2^t/\lambda_0, \sigma)$.*

Remark 4.3. We emphasize that `AdaptSmooth` requires less parameter tuning effort than the old reduction for the same reason as in Remark 3.2. Also, `AdaptSmooth`, when applied to Katyusha, provides the fastest running time on solving the Case 3 finite-sum form of (1.1), similar to Corollary 3.3.

5 JointAdaptRegSmooth: From Case 4 to 1

We show in Appendix C that `AdaptReg` and `AdaptSmooth` can work together to reduce the finite-sum form of Case 4 to Case 1. We call this reduction `JointAdaptRegSmooth` and it relies on a jointly exponentially decreasing sequence of (σ_t, λ_t) , where σ_t is the weight of the convexity parameter that we add on top of $F(x)$, and λ_t is the smoothing parameter that determines how we change each $f_i(\cdot)$. The analysis is analogous to a careful combination of the proofs for `AdaptReg` and `AdaptSmooth`.

6 Experiments

We perform experiments to confirm our theoretical speed-ups obtained for `AdaptSmooth` and `AdaptReg`. We work on minimizing Lasso and SVM objectives for the following three well-known datasets that can be found on the LibSVM website [8]: `covtype`, `mnist`, and `rcv1`. We defer some dataset and implementation details to Appendix A.

6.1 Experiments on AdaptReg

To test the performance of `AdaptReg`, consider the Lasso objective which is non-SC but smooth. We apply `AdaptReg` to reduce it to Case 1 and apply either APCG [18], an accelerated method, or (Prox-)SDCA [27, 28], a non-accelerated method. Let us make a few remarks:

- APCG and SDCA are both *indirect* solvers for non-strongly convex objectives and therefore regularization is intrinsically required in order to run them for Lasso or more generally Case 2.⁸
- APCG and SDCA do not satisfy HOOD in theory. However, they still benefit from `AdaptReg` as we shall see, demonstrating the practical value of `AdaptReg`.

A Practical Implementation. In principle, one can implement `AdaptReg` by setting the termination criteria of the oracle in the inner loop as precisely suggested by the theory, such as setting the number of iterations for SDCA to be exactly $T = O(n + \frac{L}{\sigma_t})$ in the t -th epoch. However, in

⁸Note that some other methods, such as SVRG, although only providing theoretical results for strongly convex and smooth objectives (Case 1), in practice works for Case 2 directly. Therefore, it is not needed to apply `AdaptReg` on such methods at least in practice.

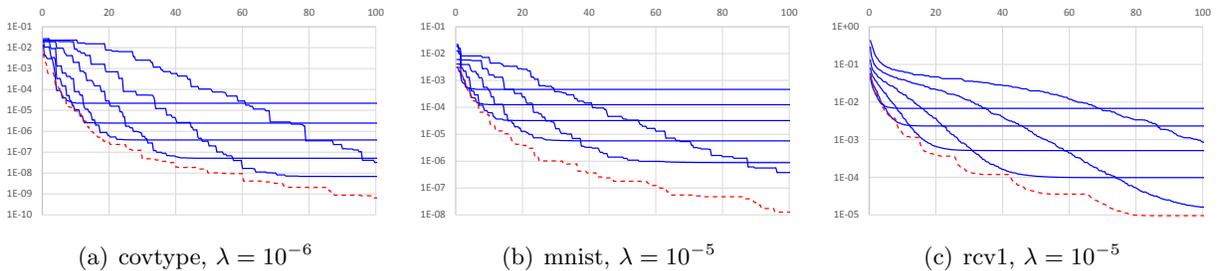


Figure 1: Comparing AdaptReg and the classical reduction on Lasso (with ℓ_1 regularizer weight λ). y -axis is the objective distance to minimum, and x -axis is the number of passes to the dataset. The blue solid curves represent APCG under the old regularization reduction, and the red dashed curve represents APCG under AdaptReg. For other values of λ , or the results on SDCA, please refer to Figure 3 and 4 in the appendix.

practice, it is more desirable to automatically terminate the oracle whenever the objective distance to the minimum has been sufficiently decreased. In all of our experiments, we simply compute the duality gap and terminate the oracle whenever the duality gap is below $1/4$ times the last recorded duality gap of the previous epoch. For details see Appendix A.

Experimental Results. For each dataset, we consider three different magnitudes of regularization weights for the ℓ_1 regularizer in the Lasso objective. This totals 9 analysis tasks for each algorithm.

For each such a task, we first implement the old reduction by adding an additional $\frac{\sigma}{2}\|x\|^2$ term to the Lasso objective and then apply APCG or SDCA. We consider values of σ in the set $\{10^k, 3 \cdot 10^k : k \in \mathbb{Z}\}$ and show the most representative six of them in the plots (blue solid curves in Figure 3 and Figure 4). Naturally, for a larger value of σ the old reduction converges faster but to a point that is *farther* from the exact minimizer because of the bias. We implement AdaptReg where we choose the initial parameter σ_0 *also* from the set $\{10^k, 3 \cdot 10^k : k \in \mathbb{Z}\}$ and present the best one in each of 18 plots (red dashed curves in Figure 3 and Figure 4). Due to space limitations, we provide only 3 of the 18 plots for medium-sized λ in the main body of this paper (see Figure 1), and include Figure 3 and 4 only in the appendix.

It is clear from our experiments that

- AdaptReg is more efficient than the old regularization reduction;
- AdaptReg requires no more parameter tuning than the classical reduction;
- AdaptReg is unbiased so simplifies the parameter selection procedure.⁹

6.2 Experiments on AdaptSmooth

To test the performance of AdaptSmooth, consider the SVM objective which is non-smooth but SC. We apply AdaptSmooth to reduce it to Case 1 and apply SVRG [12]. We emphasize that SVRG is an *indirect* solver for non-smooth objectives and therefore regularization is intrinsically required in order to run SVRG for SVM or more generally for Case 3.¹⁰

⁹It is easy to determine the best σ_0 in AdaptReg, and in contrast, in the old reduction if the desired error is somehow changed for the application, one has to select a different σ and restart the algorithm.

¹⁰Note that some other methods, such as APCG or SDCA, although only providing theoretical guarantees for strongly convex and smooth objectives (Case 1), in practice work for Case 2 directly without smoothing (see for instance the discussion in [27]). Therefore, it is unnecessary to apply AdaptSmooth to such methods at least in practice.

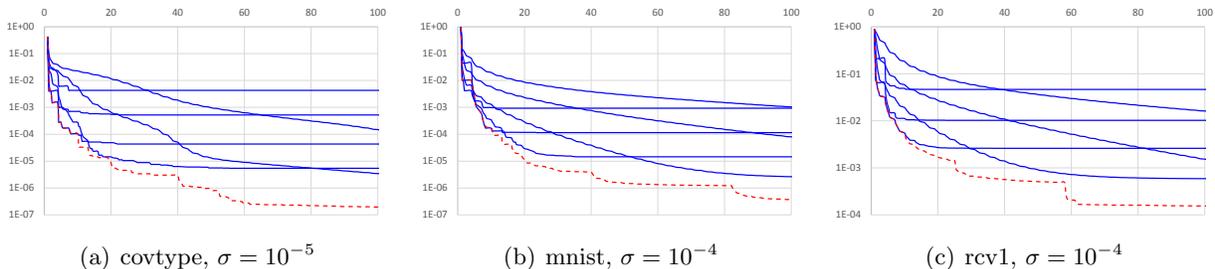


Figure 2: Comparing AdaptSmooth and the classical reduction on SVM (with ℓ_2 regularizer weight λ). y -axis is the objective distance to minimum, and x -axis is the number of passes to the dataset. The **blue solid curves** represent SVRG under the old smoothing reduction, and the **red dashed curve** represents SVRG under AdaptSmooth. For other values of σ , please refer to Figure 5 in the appendix.

A Practical Implementation. In principle, one can implement AdaptSmooth by setting the termination criteria of the oracle in the inner loop as precisely suggested by the theory, such as setting the number of iterations for SVRG to be exactly $T = O(n + \frac{1}{\sigma\lambda_t})$ in the t -th epoch. In practice, however, it is more desirable to automatically terminate the oracle whenever the objective distance to the minimum has been sufficiently decreased. In all of our experiments, we simply compute the Euclidean norm of the full gradient of the objective, and terminate the oracle whenever the norm is below $1/3$ times the last recorded Euclidean norm of the previous epoch. For details see Appendix A.

Experimental Results. For each dataset, we consider three different magnitudes of regularization weights for the ℓ_2 regularizer in the SVM objective. This totals 9 analysis tasks. For each such a task, we first implement the old reduction by smoothing the hinge loss functions (using Definition 4.1) with parameter $\lambda > 0$ and then apply SVRG. We consider different values of λ in the set $\{10^k, 3 \cdot 10^k : k \in \mathbb{Z}\}$ and show the most representative six of them in the plots (**blue solid curves** in Figure 5). Naturally, for a larger λ , the old reduction converges faster but to a point that is *farther* from the exact minimizer due to its bias. We then implement AdaptSmooth where we choose the initial smoothing parameter λ_0 also from the set $\{10^k, 3 \cdot 10^k : k \in \mathbb{Z}\}$ and present the best one in each of the 9 plots (**red dashed curves** in Figure 5). Due to space limitations, we provide only 3 of the 9 plots for small-sized σ in the main body of this paper (see Figure 2, and include Figure 5 only in the appendix.

It is clear from our experiments that

- AdaptSmooth is more efficient than the old smoothing reduction, especially when the desired training error is small;
- AdaptSmooth requires no more parameter tuning than the classical reduction;
- AdaptSmooth is unbiased so simplifies the parameter selection for the same reason as Footnote 9.

Acknowledgements

We thank Yang Yuan for very enlightening conversations, and Alon Gonen for catching a few typos in an earlier version of this paper. This paper is partially supported by a Microsoft Research Grant, no. 0518584.

References

- [1] Zeyuan Allen-Zhu. Katyusha: Accelerated Variance Reduction for Faster SGD. *ArXiv e-prints*, abs/1603.05953, March 2016.
- [2] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *ArXiv e-prints*, abs/1407.1537, July 2014.
- [3] Zeyuan Allen-Zhu, Peter Richtárik, Zheng Qu, and Yang Yuan. Even Faster Accelerated Coordinate Descent Using Non-Uniform Sampling. In *ICML*, 2016.
- [4] Zeyuan Allen-Zhu and Yang Yuan. Improved SVRG for Non-Strongly-Convex or Sum-of-Non-Convex Objectives. In *ICML*, 2016.
- [5] Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to Nesterov’s accelerated gradient descent. *ArXiv e-prints*, abs/1506.08187, June 2015.
- [6] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.
- [7] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In *NIPS*, 2014.
- [8] Rong-En Fan and Chih-Jen Lin. LIBSVM Data: Classification, Regression and Multi-label. Accessed: 2015-06.
- [9] Roy Frostig, Rong Ge, Sham M. Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *ICML*, volume 37, pages 1–28, 2015.
- [10] Elad Hazan. DRAFT: Introduction to online convex optimization. *Foundations and Trends in Machine Learning*, XX(XX):1–168, 2015.
- [11] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization. *The Journal of Machine Learning Research*, 15(1):2489–2512, 2014.
- [12] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, NIPS 2013, pages 315–323, 2013.
- [13] Simon Lacoste-Julien, Mark W. Schmidt, and Francis R. Bach. A simpler approach to obtaining an $o(1/t)$ convergence rate for the projected stochastic subgradient method. *ArXiv e-prints*, abs/1212.2002, 2012.
- [14] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *FOCS*, pages 147–156. IEEE, 2013.
- [15] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *CoRR*, abs/1408.3595, 2014.
- [16] Hongzhou Lin. private communication, 2016.

- [17] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A Universal Catalyst for First-Order Optimization. In *NIPS*, 2015.
- [18] Qihang Lin, Zhaosong Lu, and Lin Xiao. An Accelerated Proximal Coordinate Gradient Method and its Application to Regularized Empirical Risk Minimization. In *NIPS*, pages 3059–3067, 2014.
- [19] Arkadi Nemirovski. Prox-Method with Rate of Convergence $O(1/t)$ for Variational Inequalities with Lipschitz Continuous Monotone Operators and Smooth Convex-Concave Saddle Point Problems. *SIAM Journal on Optimization*, 15(1):229–251, January 2004.
- [20] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Doklady AN SSSR (translated as Soviet Mathematics Doklady)*, volume 269, pages 543–547, 1983.
- [21] Yurii Nesterov. *Introductory Lectures on Convex Programming Volume: A Basic course*, volume I. Kluwer Academic Publishers, 2004.
- [22] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, December 2005.
- [23] Francesco Orabona, Andreas Argyriou, and Nathan Srebro. Prisma: Proximal iterative smoothing algorithm. *arXiv preprint arXiv:1206.2372*, 2012.
- [24] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, 2012.
- [25] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, pages 1–45, 2013. Preliminary version appeared in NIPS 2012.
- [26] Shai Shalev-Shwartz. Online Learning and Online Convex Optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.
- [27] Shai Shalev-Shwartz and Tong Zhang. Proximal Stochastic Dual Coordinate Ascent. *arXiv preprint arXiv:1211.2717*, pages 1–18, 2012.
- [28] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013.
- [29] Shai Shalev-Shwartz and Tong Zhang. Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization. In *ICML*, pages 64–72, 2014.
- [30] Blake Woodworth and Nati Srebro. Tight Complexity Bounds for Optimizing Composite Objectives. *Working manuscript*, 2016.
- [31] Lin Xiao and Tong Zhang. A Proximal Stochastic Gradient Method with Progressive Variance Reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.
- [32] Yuchen Zhang and Lin Xiao. Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization. In *ICML*, 2015.

APPENDIX

A Experiment Details

The datasets we used in this paper are downloaded from the LibSVM website [8]:

- the covtype (binary.scale) dataset (581,012 samples and 54 features).
- the mnist (class 1) dataset (60,000 samples and 780 features).
- the rcv1 (train.binary) dataset (20,242 samples and 47,236 features).

To make easier comparison across datasets, we scale every vector by the average Euclidean norm of all the vectors in the dataset. In other words, we ensure that the data vectors have an average Euclidean norm 1. This step is for comparison only and not necessary in practice.

We use the default step-length choice for APCG which requires solving a quadratic univariate function per iteration; for SDCA, to avoid the issue for tuning step lengths, we use the steepest descent (i.e., automatic) choice which is Option I for SDCA [27]; for SVRG, we use the default step length $\eta = 1/L$.

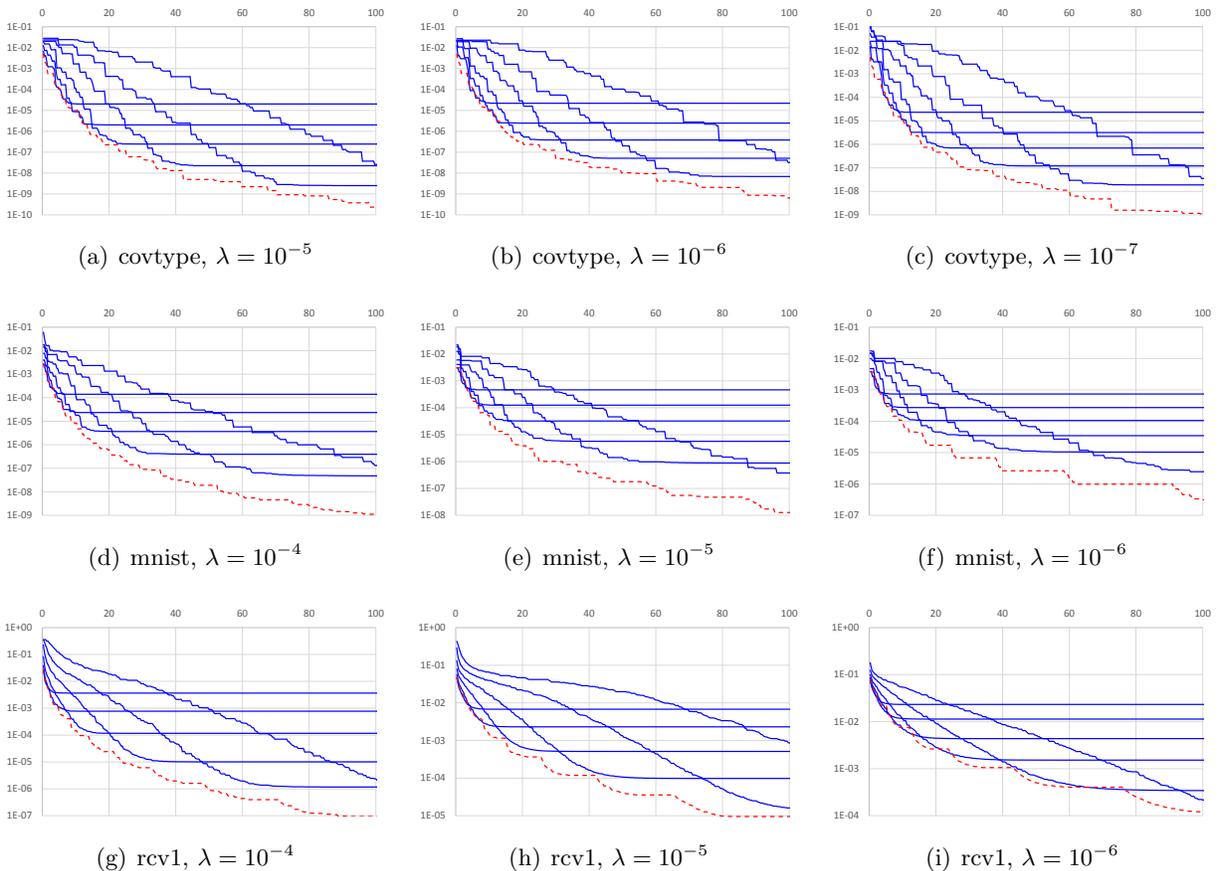


Figure 3: Performance Comparison for Lasso with weight λ on the ℓ_1 regularizer. The y axis represents the objective distance to minimum, and the x axis represents the number of passes to the dataset. The **blue solid curves** represent APCG under the old regularization reduction, and the **red dashed curve** represents APCG under AdaptReg.

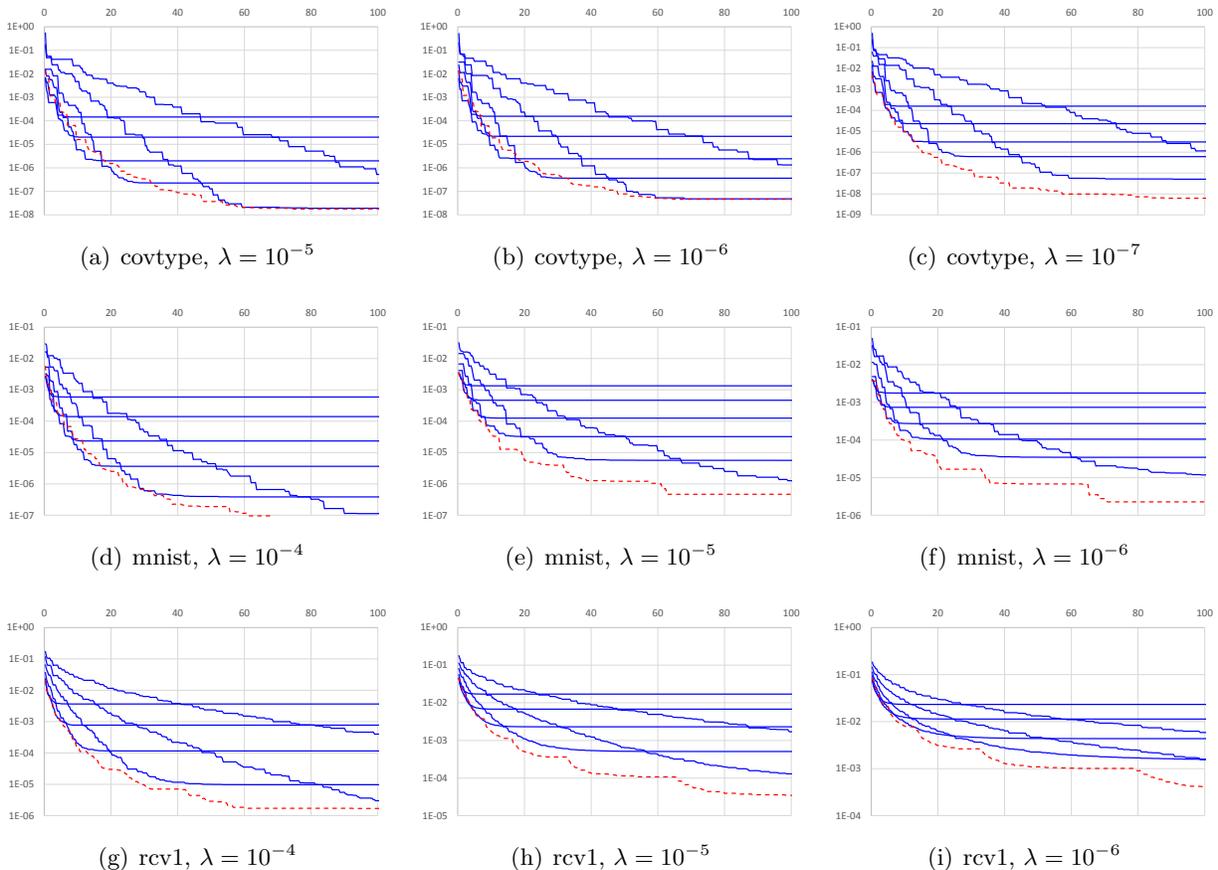


Figure 4: Performance Comparison for Lasso with weight λ on the ℓ_1 regularizer. The y axis represents the objective distance to minimum, and the x axis represents the number of passes to the dataset. The **blue solid curves** represent SDCA under the old regularization reduction, and the **red dashed curve** represents SDCA under AdaptReg.

When applying our reductions, it is desirable to automatically terminate the oracle whenever the objective distance to the minimum has been sufficiently decreased, say, by a factor of 4. Unfortunately, the oracle usually does not know the exact minimizer and cannot compute the exact objective distance to the minimum (i.e., D_t). Instead, we use the following heuristics which were also used by other reduction methods such as Catalyst [16].

- Since SDCA and APCG are primal-dual methods, in our experiments, we compute instead the duality gap which gives a reasonable approximation on D_t . More specifically, for both experiments, we compute the duality gap every $n/3$ iterations inside the implementation of APCG/SDCA, and terminate it whenever the duality gap is below $1/4$ times the last recorded duality gap of the previous epoch. Although one can further tune this parameter $1/4$ for a better performance, to perform a fair comparison, we simply set it to be identically $1/4$ across all the datasets and analysis tasks.
- When applying SVRG to Lasso, we cannot compute the duality gap because the objective is not strongly convex. In our experiments, we compute instead the Euclidean norm of the full gradient of the objective (i.e., $\|\nabla f(x)\|$) which gives a reasonable approximation on D_t . More specifically, we use the default setting of SVRG Option I that is to compute a “gradient

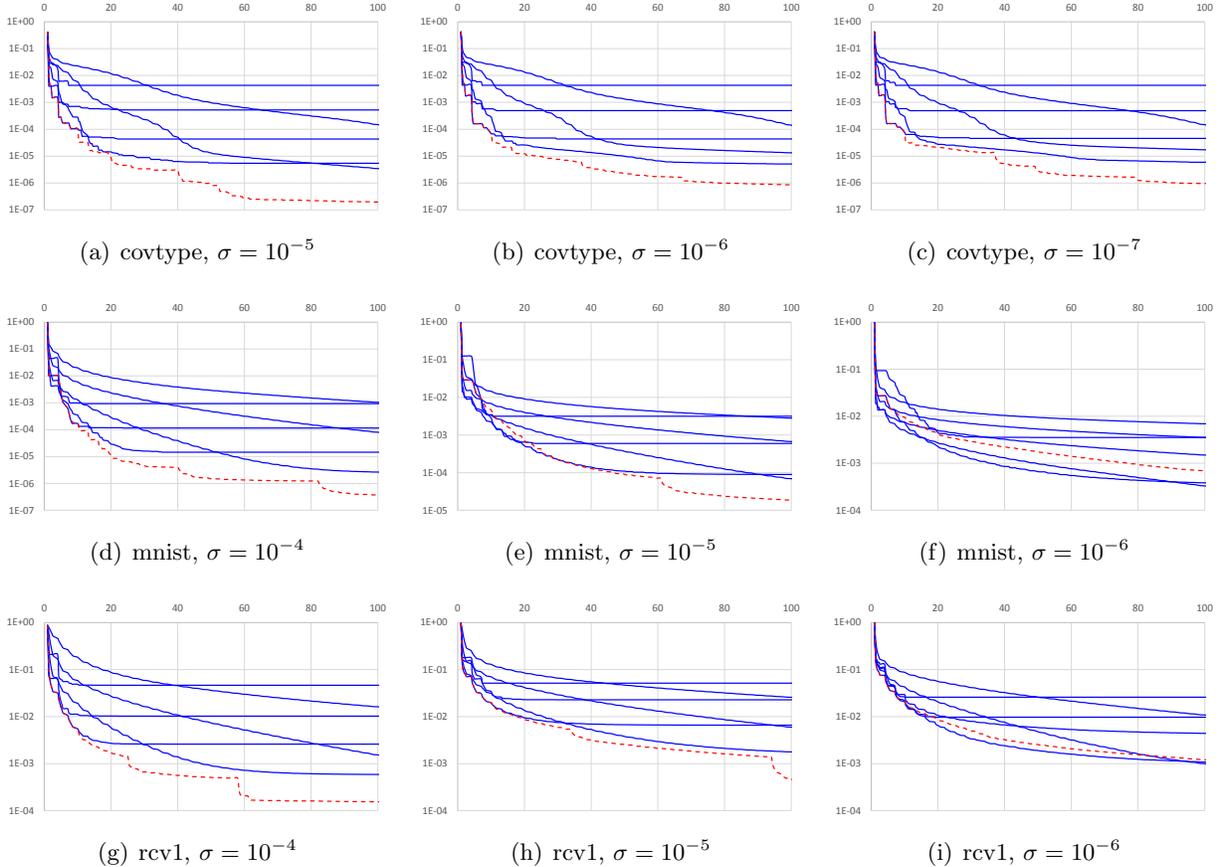


Figure 5: Performance Comparison for L2-SVM with weight σ on the ℓ_2 regularizer. The y axis represents the objective distance to minimum, and the x axis represents the number of passes to the dataset. The **blue solid curves** represent SVRG under the old smoothing reduction, and the **red dashed curve** represents SVRG under AdaptSmooth.

snapshot” every $2n$ iterations. When a gradient snapshot is computed, we can also compute its Euclidean norm almost for free. If this norm is below $1/3$ times the last norm-of-gradient of the previous epoch, we terminate SVRG for the current epoch. Note that one can further tune this parameter $1/3$ for a better performance; however, to perform a fair comparison in this paper, we simply set it to be identically $1/3$ across all the datasets and analysis tasks.

B Convergence Analysis for AdaptSmooth

We first recall the following property that bounds the difference between f_i and $f_i^{(\lambda)}$ as a function of λ :

Lemma B.1. *If each $f_i(\cdot)$ is G -Lipschitz continuous, it satisfies $f_i(\alpha) - \frac{\lambda G^2}{2} \leq f_i^{(\lambda)}(\alpha) \leq f_i(\alpha)$.*

Proof. Letting $\beta^* \stackrel{\text{def}}{=} \arg \max_{\beta} \{\beta \cdot \alpha - f_i^*(\beta)\}$, we have $\beta^* \in [-G, G]$ because the domain of $f_i^*(\cdot)$ equals the range of $\nabla f_i(\cdot)$ which is a subset of $[-G, G]$ due to the Lipschitz continuity of $f_i(\cdot)$. As

Algorithm 2 The AdaptSmooth Reduction

Input: an objective $F(\cdot)$ in finite-sum form of Case 3 (strongly convex and not necessarily smooth);
 x_0 a starting vector, λ_0 an initial smoothing parameter, T the number of epochs;
 an algorithm \mathcal{A} that solves the finite-sum form of Case 1 for problem (1.1).

Output: \hat{x}_T .

- 1: $\hat{x}_0 \leftarrow x_0$.
 - 2: **for** $t \leftarrow 0$ **to** $T - 1$ **do**
 - 3: Define $F^{(\lambda_t)}(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i^{(\lambda_t)}(\langle a_i, x \rangle) + \psi(x)$ using Definition 4.1.
 - 4: $\hat{x}_{t+1} \leftarrow \mathcal{A}(F^{(\lambda_t)}, \hat{x}_t)$.
 - 5: $\lambda_{t+1} \leftarrow \lambda_t/2$.
 - 6: **end for**
 - 7: **return** \hat{x}_T .
-

a result, we have

$$\begin{aligned} f_i(\alpha) &= \max_{\beta} \{\beta \cdot \alpha - f_i^*(\beta)\} = \beta^* \cdot \alpha - f_i^*(\beta^*) - \frac{\lambda}{2}(\beta^*)^2 + \frac{\lambda}{2}(\beta^*)^2 \\ &\leq \max_{\beta} \{\beta \cdot \alpha - f_i^*(\beta) - \frac{\lambda}{2}\beta^2\} + \frac{\lambda}{2}(\beta^*)^2 = f_i^{(\lambda)}(\alpha) + \frac{\lambda}{2}(\beta^*)^2 \leq f_i^{(\lambda)}(\alpha) + \frac{\lambda G^2}{2} . \end{aligned}$$

The other inequality is obvious. □

We also note that

Fact B.2. For $\lambda_1 \geq \lambda_2$, we have $f_i^{(\lambda_1)}(\alpha) \leq f_i^{(\lambda_2)}(\alpha)$ for every $\alpha \in \mathbb{R}$.

For analysis purpose only, we define x_{t+1} to be the exact minimizer of $F^{(\lambda_t)}(x)$. The HOOD property of the given oracle \mathcal{A} ensures that

$$F^{(\lambda_t)}(\hat{x}_{t+1}) - F^{(\lambda_t)}(x_{t+1}) \leq \frac{F^{(\lambda_t)}(\hat{x}_t) - F^{(\lambda_t)}(x_{t+1})}{4} . \quad (\text{B.1})$$

We denote by x^* the minimizer of $F(x)$, and define $\mathbf{D}_t \stackrel{\text{def}}{=} F^{(\lambda_t)}(\hat{x}_t) - F^{(\lambda_t)}(x_{t+1})$ to be the initial objective distance to the minimum on function $F^{(\lambda_t)}(\cdot)$ before we call \mathcal{A} in epoch t . At epoch 0, we simply have the upper bound

$$\mathbf{D}_0 = F^{(\lambda_0)}(x_0) - F^{(\lambda_0)}(x_1) \leq F(x_0) - F(x_1) + \frac{\lambda_0 G^2}{2} \leq F(x_0) - F(x^*) + \frac{\lambda_0 G^2}{2} .$$

Above, the first inequality is by Lemma B.1 and Fact B.2, and the second inequality is because x^* is the minimizer of $F(\cdot)$. Next, for each epoch $t \geq 1$, we compute that

$$\begin{aligned} \mathbf{D}_t &\stackrel{\text{def}}{=} F^{(\lambda_t)}(\hat{x}_t) - F^{(\lambda_t)}(x_{t+1}) \\ &\leq F^{(\lambda_{t-1})}(\hat{x}_t) + \frac{\lambda_{t-1} G^2}{2} - F^{(\lambda_{t-1})}(x_{t+1}) \\ &\leq F^{(\lambda_{t-1})}(\hat{x}_t) + \frac{\lambda_{t-1} G^2}{2} - F^{(\lambda_{t-1})}(x_t) \leq \frac{\mathbf{D}_{t-1}}{4} + \frac{\lambda_{t-1} G^2}{2} . \end{aligned}$$

Above, the first inequality is by Lemma B.1 and Fact B.2, and the second inequality is because x_t is the minimizer of $F^{(\lambda_{t-1})}(\cdot)$.

Therefore, by telescoping the above inequality and the choice $\lambda_t = \lambda_{t-1}/2$, we have that

$$D_T \leq \frac{F(x_0) - F(x^*)}{4^T} + G^2 \cdot \left(\frac{\lambda_{T-1}}{2} + \frac{\lambda_{T-2}}{8} + \dots \right) \leq \frac{F(x_0) - F(x^*)}{4^T} + 2\lambda_T G^2 .$$

In sum, we obtain a vector \hat{x}_T satisfying

$$\begin{aligned} F(\hat{x}_T) - F(x^*) &\leq F^{(\lambda_T)}(\hat{x}_T) - F^{(\lambda_T)}(x^*) + \frac{\lambda_T G^2}{2} \\ &\leq F^{(\lambda_T)}(\hat{x}_T) - F^{(\lambda_T)}(x_{T+1}) + \frac{\lambda_T G^2}{2} \\ &= D_T + \frac{\lambda_T G^2}{2} \leq \frac{1}{4^T} (F(x_0) - F(x^*)) + 2.5\lambda_T G^2 \end{aligned} \quad (\text{B.2})$$

Finally, after appropriately choosing λ_0 and T , (B.2) directly implies Theorem 4.2.

C JointAdaptRegSmooth: Reduction from Case 4 to Case 1

In this section, we show that AdaptReg and AdaptSmooth can work together to solve the finite-sum form of Case 4. That is,

$$\min_x F(x) = \frac{1}{n} \sum_{i=1}^n f_i(\langle a_i, x \rangle) + \psi(x) ,$$

where $\psi(x)$ is not necessarily strongly convex and each $f_i(\cdot)$ may not be smooth (but is Lipschitz continuous). Without loss of generality, we assume $\|a_i\| = 1$ for each $i \in [n]$. We solve this problem by reducing it to an algorithm \mathcal{A} solving the finite-sum form of Case 1 that satisfies HOOD.

Following the same definition of $f_i^{(\lambda)}(\cdot)$ in Definition 4.1, in this section, we consider the following regularized smoothed objective $F^{(\lambda, \sigma)}(x)$:

Definition C.1. *Given parameters $\lambda, \sigma > 0$, let*

$$F^{(\lambda, \sigma)}(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i^{(\lambda)}(\langle a_i, x \rangle) + \psi(x) + \frac{\sigma}{2} \|x - x_0\|^2 .$$

From this definition we know that $F^{(\lambda, \sigma)}(x)$ falls into the finite-sum form of Case 1 for problem (1.1) with $L = 1/\lambda$ and σ being the strong convexity parameter.

JointAdaptRegSmooth works as follows (see Algorithm 3). At the beginning of the reduction, we set \hat{x}_0 to equal x_0 , an arbitrary given starting vector. JointAdaptRegSmooth consists of T epochs. At each epoch $t = 0, 1, \dots, T-1$, we define a $(1/\lambda_t)$ -smooth σ_t -strongly convex objective $F^{(\lambda_t, \sigma_t)}(x)$ using Definition C.1 above. Here, the parameters $\lambda_{t+1} = \lambda_t/2$ and $\sigma_{t+1} = \sigma_t/2$ for each $t \geq 0$, and λ_0, σ_0 are two input parameters to JointAdaptRegSmooth that will be specified later. We run \mathcal{A} on $F^{(\lambda_t, \sigma_t)}(x)$ with starting vector \hat{x}_t in each epoch, and let the output be \hat{x}_{t+1} . After all T epochs are finished, JointAdaptRegSmooth simply outputs \hat{x}_T . (Alternatively, if one sets T to be infinity, JointAdaptRegSmooth can be interrupted at an arbitrary moment and output \hat{x}_t of the current epoch.)

We state our main theorem for JointAdaptRegSmooth below and prove it in Section D.

Theorem C.2. *Suppose that in problem (1.1), each $f_i(\cdot)$ is G -Lipschitz continuous. Let x_0 be a starting vector such that $F(x_0) - F(x^*) \leq \Delta$ and $\|x_0 - x^*\|^2 \leq \Theta$. Then, JointAdaptRegSmooth with $\lambda_0 = \Delta/G^2$, $\sigma_0 = \Delta/\Theta$ and $T = \log_2(\Delta/\varepsilon)$ produces an output \hat{x}_T satisfying $F(\hat{x}_T) - \min_x F(x) \leq O(\varepsilon)$ in a total running time of $\sum_{t=0}^{T-1} \text{Time}(2^t/\lambda_0, \sigma_0 \cdot 2^{-t})$.*

Algorithm 3 The JointAdaptRegSmooth Reduction

Input: an objective $F(\cdot)$ in finite-sum form of Case 4 (not necessarily strongly convex or smooth);
 x_0 starting vector, λ_0, σ_0 initial smoothing and regularization params, T number of epochs;
an algorithm \mathcal{A} that solves the finite-sum form of Case 1 for problem (1.1).

Output: \hat{x}_T .

- 1: $\hat{x}_0 \leftarrow x_0$.
 - 2: **for** $t \leftarrow 0$ **to** $T - 1$ **do**
 - 3: Define $F^{(\lambda_t, \sigma_t)}(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i^{(\lambda_t, \sigma_t)}(\langle a_i, x \rangle) + \psi(x) + \frac{\sigma_t}{2} \|x - x_0\|^2$ using Definition C.1.
 - 4: $\hat{x}_{t+1} \leftarrow \mathcal{A}(F^{(\lambda_t)}, \hat{x}_t)$.
 - 5: $\sigma_{t+1} \leftarrow \sigma_t/2, \lambda_{t+1} \leftarrow \lambda_t/2$.
 - 6: **end for**
 - 7: **return** \hat{x}_T .
-

Example C.3. When JointAdaptRegSmooth is applied to an accelerated gradient descent method such as [2, 5, 14, 15, 20–22], we solve the finite-sum form of Case 4 with a total running time $\sum_{t=0}^{T-1} \text{Time}(2^t/\lambda_0, \sigma_0 \cdot 2^{-t}) = O(\text{Time}(1/\lambda_T, \sigma_T)) = O(G\sqrt{\Theta}/\varepsilon) \cdot C$. This matches the best known running time of full-gradient first-order methods on solving Case 4, which usually is obtained via saddle-point based methods such as Chambolle-Pock [6] or the mirror prox method of Nemirovski [19].

D Convergence Analysis for JointAdaptRegSmooth

For analysis purpose only, we define x_{t+1} to be the exact minimizer of $F^{(\lambda_t, \sigma_t)}(x)$. The HOOD property of the given oracle \mathcal{A} ensures that

$$F^{(\lambda_t, \sigma_t)}(\hat{x}_{t+1}) - F^{(\lambda_t, \sigma_t)}(x_{t+1}) \leq \frac{F^{(\lambda_t, \sigma_t)}(\hat{x}_t) - F^{(\lambda_t, \sigma_t)}(x_{t+1})}{4} .$$

We denote by x^* an arbitrary minimizer of $F(x)$. The following claim states a simple property about the minimizers of $F^{(\lambda_t, \sigma_t)}(x)$ which is analogous to Claim 3.4:

Claim D.1. *We have $\frac{\sigma_t}{2} \|x_{t+1} - x^*\|^2 \leq \frac{\sigma_t}{2} \|x_0 - x^*\|^2 + \frac{\lambda_t G^2}{2}$ for each $t \geq 0$.*

Proof. By the strong convexity of $F^{(\lambda_t, \sigma_t)}(x)$ and the fact that x_{t+1} is its exact minimizer, we have

$$F^{(\lambda_t, \sigma_t)}(x_{t+1}) - F^{(\lambda_t, \sigma_t)}(x^*) \leq -\frac{\sigma_t}{2} \|x_{t+1} - x^*\|^2 .$$

Using the fact that $F^{(\lambda_t, \sigma_t)}(x_{t+1}) \geq F^{(\lambda_t, 0)}(x_{t+1}) \geq F(x_{t+1}) + \frac{\lambda_t G^2}{2}$ (where the second inequality follows from Lemma B.1), as well as the definition $F^{(\lambda_t, \sigma_t)}(x^*) = F^{(\lambda_t, 0)}(x^*) + \frac{\sigma_t}{2} \|x^* - x_0\|^2 \leq F(x^*) + \frac{\sigma_t}{2} \|x^* - x_0\|^2$ (where the second inequality again follows from Lemma B.1), we immediately have

$$\frac{\lambda_t G^2}{2} + \frac{\sigma_t}{2} \|x_0 - x^*\|^2 - \frac{\sigma_t}{2} \|x_{t+1} - x^*\|^2 \geq F(x_{t+1}) - F(x^*) \geq 0 . \quad \square$$

Let $D_t \stackrel{\text{def}}{=} F^{(\lambda_t, \sigma_t)}(\hat{x}_t) - F^{(\lambda_t, \sigma_t)}(x_{t+1})$ be the initial objective distance to the minimum on function $F^{(\lambda_t, \sigma_t)}(\cdot)$ before we call \mathcal{A} in epoch t . At epoch 0, we simply have the upper bound

$$\begin{aligned} D_0 &= F^{(\lambda_0, \sigma_0)}(x_0) - F^{(\lambda_0, \sigma_0)}(x_1) \stackrel{\textcircled{1}}{\leq} F^{(0, \sigma_0)}(x_0) - F^{(\lambda_0, 0)}(x_1) \\ &\stackrel{\textcircled{2}}{\leq} F(x_0) - F(x_1) + \frac{\lambda_0 G^2}{2} \stackrel{\textcircled{3}}{\leq} F(x_0) - F(x^*) + \frac{\lambda_0 G^2}{2} . \end{aligned}$$

Above, ① uses $F^{(\lambda_0, \sigma_0)}(x_0) \leq F^{(0, \sigma_0)}(x_0)$ which is a consequence of Fact B.2; ② uses $F^{(0, \sigma_0)}(x_0) = F(x_0)$ from the definition and $F(x_1) \leq F^{(\lambda_0, 0)}(x_1) + \frac{\lambda_0 G^2}{2}$ from Lemma B.1; and ③ uses the minimality of x^* . Next, for each epoch $t \geq 1$, we compute that

$$\begin{aligned}
D_t &\stackrel{\text{def}}{=} F^{(\lambda_t, \sigma_t)}(\hat{x}_t) - F^{(\lambda_t, \sigma_t)}(x_{t+1}) \\
&\stackrel{\text{①}}{\leq} F^{(\lambda_t, \sigma_{t-1})}(\hat{x}_t) - F^{(\lambda_t, \sigma_{t-1})}(x_{t+1}) + \frac{\sigma_{t-1} - \sigma_t}{2} \|x_{t+1} - x_0\|^2 \\
&\stackrel{\text{②}}{\leq} F^{(\lambda_{t-1}, \sigma_{t-1})}(\hat{x}_t) + \frac{\lambda_{t-1} G^2}{2} - F^{(\lambda_{t-1}, \sigma_{t-1})}(x_{t+1}) + \frac{\sigma_t}{2} \|x_{t+1} - x_0\|^2 \\
&\stackrel{\text{③}}{\leq} F^{(\lambda_{t-1}, \sigma_{t-1})}(\hat{x}_t) + \frac{\lambda_{t-1} G^2}{2} - F^{(\lambda_{t-1}, \sigma_{t-1})}(x_{t+1}) + \sigma_t \|x_{t+1} - x^*\|^2 + \sigma_t \|x_0 - x^*\|^2 \\
&\stackrel{\text{④}}{\leq} F^{(\lambda_{t-1}, \sigma_{t-1})}(\hat{x}_t) + \frac{\lambda_{t-1} G^2}{2} - F^{(\lambda_{t-1}, \sigma_{t-1})}(x_t) + 2\sigma_t \|x_0 - x^*\|^2 + \lambda_t G^2 \\
&\stackrel{\text{⑤}}{\leq} \frac{D_{t-1}}{4} + 2\sigma_t \|x_0 - x^*\|^2 + 2\lambda_t G^2 .
\end{aligned}$$

Above, ① follows from the definition; ② follows from Lemma B.1, Fact B.2 as well as the choice $\sigma_{t-1} = 2\sigma_t$; ③ follows because for any two vectors a, b it satisfies $\|a - b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$; ④ follows from Claim D.1; ⑤ follows from the definition of D_{t-1} , from (3.1), and from the choice $\lambda_{t-1} = 2\lambda_t$.

By telescoping the above inequality, we have

$$\begin{aligned}
D_T &\leq \frac{F(x_0) - F(x^*)}{4^T} + G^2 \cdot \left(2\lambda_T + \frac{2\lambda_{T-1}}{4} + \dots\right) + \|x_0 - x^*\|^2 \cdot \left(2\sigma_T + \frac{2\sigma_{T-1}}{4} + \dots\right) \\
&\leq \frac{1}{4^T} (F(x_0) - F(x^*)) + 4\lambda_T G^2 + 4\sigma_T \|x_0 - x^*\|^2 ,
\end{aligned}$$

where the second inequality uses our choice $\lambda_t = \lambda_{t-1}/2$ and $\sigma_t = \sigma_{t-1}/2$ again. In sum, we obtain a vector \hat{x}_T satisfying

$$\begin{aligned}
F(\hat{x}_T) - F(x^*) &\stackrel{\text{①}}{\leq} F^{(\lambda_T, 0)}(\hat{x}_T) - F^{(0, \sigma_T)}(x^*) + \frac{\lambda_T G^2}{2} + \frac{\sigma_T}{2} \|x_0 - x^*\|^2 \\
&\stackrel{\text{②}}{\leq} F^{(\lambda_T, \sigma_T)}(\hat{x}_T) - F^{(\lambda_T, \sigma_T)}(x^*) + \frac{\lambda_T G^2}{2} + \frac{\sigma_T}{2} \|x_0 - x^*\|^2 \\
&\stackrel{\text{③}}{\leq} F^{(\lambda_T, \sigma_T)}(\hat{x}_T) - F^{(\lambda_T, \sigma_T)}(x_{T+1}) + \frac{\lambda_T G^2}{2} + \frac{\sigma_T}{2} \|x_0 - x^*\|^2 \\
&\stackrel{\text{④}}{\leq} \frac{1}{4^T} (F(x_0) - F(x^*)) + 4.5\lambda_T G^2 + 4.5\sigma_T \|x_0 - x^*\|^2 . \tag{D.1}
\end{aligned}$$

Above, ① uses Lemma B.1 and the definition; ② uses the monotonicity and Fact B.2, ③ uses the definition that x_{T+1} is the minimizer of $F^{(\lambda_T, \sigma_T)}(\cdot)$; and ④ uses the definition of D_T and our derived upper bound.

Finally, after appropriately choosing σ_0 , λ_0 and T , (D.1) immediately implies Theorem C.2.