NORTHWESTERN UNIVERSITY

Methods for Nonsmooth and Stochastic Optimization for Machine
Learning

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Industrial Engineering and Management Sciences

By

Stefan Solntsev

EVANSTON, ILLINOIS

December 2015

ABSTRACT

Methods for Nonsmooth and Stochastic Optimization for Machine Learning

Stefan Solntsev

Stochasticity and non-smoothness are two major challenges for optimization algorithms in convex optimization. This document contains descriptions of two projects, each designed to tackle one of these challenges. We first present an active-set method for quadratic ℓ_1 regularized but deterministic optimization problems. It is a novel method with unorthodox termination criteria for the subspace minimization phase. The second project focuses on creating optimization methods for stochastic but smooth problems, with special emphasis on machine learning applications. We propose an algorithm for minimizing expected risk that shares some properties with randomized incremental aggregated gradient methods as well as dynamic sampling methods. Theoretical convergence analysis is presented, and numerical results on machine learning test problems illustrate the performance of the method.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

The work described in this document consists of two parts. In the first part, we present an active-set method for minimizing an objective that is the sum of a convex quadratic and ℓ_1 regularization term. Unlike two-phase methods that combine a first-order active set identification step and a subspace phase consisting of a cycle of conjugate gradient iterations, the method presented here has the flexibility of computing one of three possible steps at each iteration: a relaxation step (that releases variables from the active set), a subspace minimization step based on the conjugate gradient iteration, and an active-set refinement step. The choice of step depends on the relative magnitudes of the components of the minimum norm subgradient. We show the global convergence properties of the algorithm, as well as work complexity estimates. Extensive testing to evaluate the efficiency of the method relative to state of the art algorithms is conducted.

In the second part of the document we propose an algorithm for minimizing expected risk F that shares some properties with randomized incremental aggregated gradient methods as well as dynamic sampling methods. Unlike aggregated gradient methods, which are designed to revisit the training set many times, the algorithm proposed here is well suited for problems involving a very large training set, where one (or a few) passes over the data suffice to produce an acceptable solution. At every iteration, the algorithm updates a collection of gradients from certain past iterations, and as in dynamic sample methods *additional* gradients are evaluated at the current point. By allowing the amount

of information to increase at every iteration the algorithm is able to achieve linear convergence in expected risk F (not just in training error). Numerical results on machine learning test problems illustrate the performance of the method.

The two projects of this thesis address topics of current interest in signal processing, statistics, machine learning, and many other fields. The minimization of a regularized quadratic function is of interest in its own right, and also appears as a subproblem in general convex regularized optimization problems. Stochastic optimization problems addressed in the second part of the dissertation arise in the number of areas such as simulation optimization and machine learning, and our algorithms and analysis will be of wide applicability.

CHAPTER 2

An Algorithm for Quadratic ℓ_1-Regularized Optimization

We present an active-set method for the solution of the regularized quadratic problem

$$(2.1) \qquad \min_{x \in \mathbb{R}^n} F(x) \triangleq \tfrac{1}{2} x^T A x - b^T x + \tau \|x\|_1,$$

where A is a symmetric positive semi-definite matrix and $\tau \geq 0$ is a regularization parameter. The motivation for this work stems from the numerous applications in signal processing, machine learning, and statistics that require the solution of problem (2.1); see e.g. [**51, 29, 50**] and the references therein.

Although non-differentiable, the quadratic-ℓ_1 problem (2.1) has a simple structure that can be exploited effectively in the design of algorithms, and in their analysis. Our focus in this chapter is on methods that incorporate second-order information about the objective function $F(x)$ as an integral part of the iteration. A salient feature of our method is the flexibility of switching between two types of steps: a) first-order steps that improve the active-set prediction; b) subspace steps that explore the current active set through an inner conjugate gradient (CG) iteration. The choice between these steps is controlled by the *gradient balance condition* that compares the norm of the free (non-zero) components of the minimum norm subgradient of F with the norm of the components corresponding to the zero variables (appropriately scaled). This condition is motivated by the work of Dostal and Schoeberl [**17**] on the solution of bound constrained problems, but in extending

the idea to the quadratic-ℓ_1 problem (2.1), we deviate from their approach in a significant way.

We present two variants of our approach that differ in the active set identification step. One variant employs the iterative soft-thresholding algorithm, ISTA [11, 16, 54], while the other computes an ISTA step on the subspace of zero variables. Our numerical tests show that the two algorithms perform efficiently compared to state-of-the-art codes. We provide global rates of convergence as well as work-complexity estimates that bound the total amount of computation needed to achieve an ϵ-accurate solution. We refer to our approach as the "interleaved ISTA-CG method", or iiCG.

The quadratic-ℓ_1 problem (2.1) has received considerable attention in the literature, and a variety of first and second order methods have been proposed for solving it. Most prominent are variants of the ISTA algorithm and its accelerated versions [36, 3, 4], which have extensive theory and are popular in practice. The TFOCS package [4] provides five first-order methods based on proximal gradient iterations that enjoy optimal complexity bounds; i.e., they achieve ϵ accuracy in at most $O(1/\sqrt{\epsilon})$ iterations. One of these methods, N83, is tested in our numerical experiments. Other first-order methods for problem (2.1) include LARS [19], coordinate descent [23], a fixed point continuation method [27], and a gradient projection method [20].

Schmidt [47] proposed several scaled sub-gradient methods, which can be viewed as extensions, to quadratic-ℓ_1 problem (2.1), of a projected quasi-Newton method [2], an active-set method [42], and a two-metric projection method [26] for bound constrained optimization. He compares these methods with some first-order methods such as GSPR [20]

and SPARSA [**54**]. We include the best-performing method (PSSgb) in our numerical tests.

Other second order methods have been proposed as well; they compute a step by minimizing a local quadratic model of F. Some of these algorithms transform problem (2.1) into a smooth bound constrained quadratic programming problem and apply an interior point procedure [**32**] or a second order gradient projection algorithm [**48, 20**]. Methods that are closer in spirit to our approach include FPC_AS [**52**], orthant-based Newton-CG methods [**2, 8, 40**], and the semi-smooth Newton method in [**35**]. Our method differs from all these approaches in the adaptive step-by-step nature of the algorithm, where a different kind of step can be invoked at every iteration. This gives the algorithm the flexibility to adapt itself to the characteristics of the problem to be solved, as we discuss in our numerical tests.

The chapter is organized in six sections. In Section 2.1 we motivate our approach and describe the first algorithm. Section 2.2 presents the second algorithm. A convergence analysis and a work complexity estimate of the two variants is given in Section 2.3. Section 2.4 describes implementation details, such as the use of a line search in the identification phase, and numerical results. The contributions of the project are summarized in Section 2.5.

2.1. The First Algorithm: iiCG-1

Let us define

$$f(x) \triangleq \tfrac{1}{2}x^T A x - b^T x, \quad \text{and} \quad g(x) \triangleq \nabla f(x) = Ax - b,$$

so that

$$F(x) = f(x) + \tau \|x\|_1.$$

Throughout the chapter, we assume that τ is fixed and has been chosen to achieve some desirable properties of the solution of the problem.

The algorithm starts by computing a first-order active-set identification step. Then, a subspace minimization step is computed over the space of free variables (i.e. the non-zero variables given by the first-order step) using the conjugate gradient (CG) method. After each CG iteration, the algorithm determines whether to continue the subspace minimization or perform a first-order step. This decision is based on the so-called gradient balance condition that we now describe.

The iterative soft-thresholding (ISTA) [**16, 11**] algorithm generates iterates as follows:

$$(2.1) \quad x^{k+1} = \arg\min_y m^k(y) \triangleq \arg\min_y f(x^k) + (y - x^k)^T g(x^k) + \frac{1}{2\alpha}\|y - x^k\|^2 + \tau\|y\|_1,$$

where $\alpha > 0$ a steplength parameter. Since m^k is a separable function, we can write it as

$$(2.2) \qquad\qquad x^{k+1} = x^k - \alpha\omega(x^k) - \alpha\psi(x^k),$$

where, for $i = 1, \ldots, n$,

$$\omega(x^k)_i \triangleq \begin{cases} 0 & \text{if} \quad x_i^k \neq 0 \\ \frac{1}{\alpha}(x_i^k - \arg\min_{y_i} m^k(y)) & \text{if} \quad x_i^k = 0 \end{cases}$$

$$(2.3) \quad = \begin{cases} 0 & \text{if} \quad x_i^k \neq 0 \\ 0 & \text{if} \quad x_i^k = 0 \text{ and } |g_i(x^k)| \leq \tau \\ g_i(x^k) - \tau \operatorname{sgn}(g_i(x^k)) & \text{if} \quad x_i^k = 0 \text{ and } |g_i(x^k)| > \tau \end{cases},$$

$$\psi(x^k)_i \triangleq \begin{cases} \frac{1}{\alpha}(x_i^k - \arg\min_{y_i} m^k(y)) & \text{if} \quad x_i^k \neq 0 \\ 0 & \text{if} \quad x_i^k = 0 \end{cases}$$

$$(2.4) \quad = \begin{cases} \frac{1}{\alpha}\left(x_i^k - \max\{|x_i^k - \alpha g_i(x^k)| - \alpha\tau, 0\} \operatorname{sgn}(x_i^k - \alpha g_i(x^k))\right) & \text{if} \quad x_i^k \neq 0 \\ 0 & \text{if} \quad x_i^k = 0 \end{cases}.$$

Following Dostal and Schoeberl [**17**], we use the magnitudes of the vectors ω and ψ to determine which type of step should be taken. The vector $\omega(x^k)$ consists of the components of the minimum norm subgradient of F corresponding to variables at zero (see (2.11)), and its norm provides a first-order estimate of the expected decrease in the objective resulting from changing those variables. Similarly, $\|\psi(x^k)\|$ provides such an estimate for the free variables, but it is more complex due to the effect of variables changing sign.

Thus, when the magnitude of $\omega(x^k)$ is large, it is an indication that releasing some of the zero variables can produce substantial improvements in the objective. On the other hand, when $\|\psi(x^k)\|$ is larger than $\|\omega(x^k)\|$, it is an indication that a move in the non-zero

variables is more beneficial. The algorithm thus monitors the *gradient balance condition*

$$(2.5) \qquad \|\omega(x^k)\|_2 \le \|\psi(x^k)\|_2,$$

which governs the flow of the iteration and distinguishes it from both two-phase methods [2, 8, 54] and semi-smooth Newton methods [35] for problem (2.1).

Let us describe the algorithm in more detail. The first order step is computed by the ISTA iteration (2.2); see e.g. [8]. The choice of the parameter α is discussed below.

The subspace minimization procedure uses the conjugate gradient method to reduce a model of the objective $F(x)$ on the subspace

$$H = \{x \mid x_i = 0, \quad \text{for all } i \text{ such that } x_i^{\text{cg}} = 0\},$$

where x^{cg} denotes the point at which the CG procedure was started (this point is provided by the ISTA step). The conjugate gradient method is applied to a smooth quadratic function q (as in [52]) that equals the objective F on the current orthant defined by x^{cg}, i.e.,

$$(2.6) \qquad q(x; x^{\text{cg}}) \triangleq \tfrac{1}{2}x^T A x + (-b + \tau \operatorname{sgn}(x^{\text{cg}}))^T x,$$

where we use the convention $\operatorname{sgn}(0) = 0$ and the fact that

$$(2.7) \qquad \|x\|_1 = \operatorname{sgn}(x)^T x.$$

Clearly, $F(x) = q(x; x^{cg})$ for all x such that $\text{sgn}(x) = \text{sgn}(x^{cg})$. The algorithm applies the projected CG iteration [**39**, chap 16] to the problem

(2.8)
$$\min_x \ q(x; x^{cg})$$

$$\text{s.t.} \ \ x \in H.$$

The gradient of q at x^k on the subspace H, is given by $P(g(x^k) + \tau \, \text{sgn}(x^k))$, where P is the projection onto H. It follows that an iteration of the projected CG method is given by

$$x^{k+1} = x^k + \alpha_{cg} d^k, \quad \text{with} \quad \alpha_{cg} = \frac{(r^k)^T \rho^k}{(d^k)^T A d^k};$$

$$r^{k+1} = r^k + \alpha_{cg} A d^k;$$

$$\rho^{k+1} = P(r^{k+1});$$

$$d^{k+1} = -\rho^{k+1} + \frac{(r^{k+1})^T \rho^{k+1}}{(r^k)^T \rho^k} d^k,$$

where initially $r^k = g(x^k) + \tau \, \text{sgn}(x^k)$, $\rho^k = P(r^k)$, $d^k = -\rho^k$.

The gradient balance condition (2.5) is tested after every CG iteration, and if it is not satisfied, the CG loop is terminated. This is a sign that substantial improvements in the objective value can be achieved by releasing some of the zero variables. This loop is also terminated when the CG iteration has crossed orthants and a sufficient reduction in the objective F was not achieved.

A precise description of the method for solving problem (2.1) is given in Algorithm iiCG-1. Here and henceforth $\|\cdot\|$ stands for the ℓ_2 norm, and $v(x)$ denotes the minimum norm subgradient of F at x.

Algorithm 1 Algorithm iiCG-1

Require: A, b, τ, x^0, c, and α
1:   $k = 0$
2:   **loop**
3:      $x^{k+1} = x^k - \alpha\omega(x^k) - \alpha\psi(x^k)$ *ISTA step*
4:      $k = k + 1$
5:      $r^k = g(x^k) + \tau \operatorname{sgn}(x^k)$, $\rho^k = P(r^k)$, $d^k = -\rho^k$, $x^{\mathrm{cg}} = x^k$
6:      **loop**
7:          **if** $\|\omega(x^k)\| > \|\psi(x^k)\|$ **then**
8:              **break**
9:          **end if**
10:         $x^{k+1} = x^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} d^k$ *CG step*
11:         $r^{k+1} = r^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} A d^k$
12:         **if** $\operatorname{sgn}(x^{k+1}) \neq \operatorname{sgn}(x^{\mathrm{cg}})$ and $F(x^{k+1}) > F(x^k) - c\|v(x^k)\|^2$ **then**
13:             $x^{k+1} = \texttt{cutback}(x_k, x^{\mathrm{cg}}, d^k)$
14:             $k = k + 1$
15:             **break**
16:         **end if**
17:         $\rho^{k+1} = P(r^{k+1})$
18:         $d^{k+1} = -\rho^{k+1} + \frac{(r^{k+1})^T \rho^{k+1}}{(r^k)^T \rho^k} d^k$
19:         $k = k + 1$
20:      **end loop**
21: **end loop**

A common choice for the stepsize is $\alpha = 1/L$ (where L is the largest eigenvalue of A) and is motivated by the convergence analysis in Section 2.3. In practice, precise knowledge of L is not needed; in Section 2.4 we discuss a heuristic way of choosing α.

Let us consider Step 13. It can be beneficial to allow the CG iteration to leave the current orthant, as long as the objective F is reduced sufficiently after every CG step.

Inspired by the analysis given in Section 2.3 (see Lemma 2.3.4), we require that

$$(2.9) \qquad\qquad F(x^{k+1}) \leq F(x^k) - c\|v(x^k)\|^2,$$

for some $c \geq 0$, where $v(x^k)$ is the minimum norm subgradient of F at x^k. The CG iteration is thus terminated as follows. If x^{k+1} is the first CG iterate that leaves the orthant, then we either accept it, if it produces the sufficient decrease (2.9) in F, or we cut it back to the boundary of the current orthant. On the other hand, if both x^{k+1} and x^k lie outside the current orthant and if sufficient decrease is not obtained at x^{k+1}, then the algorithm reverts to x^k. This procedure is given as follows:

cutback$(x_k, x^{\mathrm{cg}}, d^k)$

if $\mathrm{sgn}(x^k) = \mathrm{sgn}(x^{\mathrm{cg}})$ **then**

$\qquad \alpha_b = \arg\max_{\alpha_b}\{\alpha_b : \mathrm{sgn}(x^k + \alpha_b d^k) = \mathrm{sgn}(x^{\mathrm{cg}})\}$

$\qquad x^{k+1} = x^k + \alpha_b d^k$

else

$\qquad x^{k+1} = x^k$

end if

One of the main benefits of allowing the CG iteration to move across orthant boundaries is that this strategy prevents the generation of unnecessarily short subspace steps. In addition, if the quadratic model (2.6) does not change much as orthants change (for example, when τ is small), CG steps can be beneficial in spite of the fact that they are based on information from another orthant.

Note that in algorithm iiCG-1 the index k may be incremented multiple times during every outer iteration loop. While it is possible to express the same algorithmic logic in a

more conventional way, with a single k increment in each outer iteration, our description emphasizes that every inner CG step and ISTA step require approximately the same amount of computational effort, which is dominated by a matrix-vector product.

The algorithm of Dostal and Schoeberl includes a step along the direction $-\omega(x^k)$; its goal is release variables when the gradient balance condition does not hold. We have dispensed with this step, as we have observed that it is more effective to release variables through the ISTA iteration.

2.2. The Second Algorithm: iiCG-2

This method is motivated by the observation that the ISTA step (2.2) often releases too many zero variables. To prevent this, we replace it (under certain conditions) by a *subspace ISTA step* given by

$$(2.1) \qquad\qquad x^{k+1} = x^k - \alpha\psi(x^k).$$

Here, zero variables are kept fixed and the rest of the variables are updated by the ISTA iteration; see definition (2.4). Thus, (2.1) refines the estimate of the active set without releasing any variables.

The subspace ISTA step (2.1) is performed only when the gradient balance condition (2.5) is satisfied. If (2.5) is not satisfied, releasing some of the zero variables may be beneficial, and the full-space ISTA step (2.2) is taken to allow this. The freedom to choose among two types of active-set prediction steps provides the algorithm with a powerful active set identification mechanism; see the results in Section 2.4. The choice of the

steplength α in (2.1) is discussed in Section 2.4. A detailed description this method is given in algorithm iiCG-2.

Algorithm 2 Algorithm iiCG-2

Require: A, b, τ, x^0, c, and α

1: $k = 0$
2: **loop**
3:     **if** $\|\omega(x^k)\|^2 \leq \|\psi(x^k)\|^2$ **then**
4:         $x^{k+1} = x^k - \alpha\psi(x^k)$ *Subspace ISTA step*
5:     **else**
6:         $x^{k+1} = x^k - \alpha\omega(x^k) - \alpha\psi(x^k)$ *ISTA step*
7:     **end if**
8:     $k = k + 1$
9:     $r^k = g(x^k) + \tau\,\mathrm{sgn}(x^k)$, $\rho^k = P(r^k)$, $d^k = -\rho^k$, $x^{\mathrm{cg}} = x^k$
10:     **loop**
11:         **if** $\|\omega(x^k)\|^2 > \|\psi(x^k)\|^2$ **then**
12:             **break**
13:         **end if**
14:         $x^{k+1} = x^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} d^k$ *CG step*
15:         $r^{k+1} = r^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} A d^k$
16:         **if** $\mathrm{sgn}(x^{k+1}) \neq \mathrm{sgn}(x^{\mathrm{cg}})$ and $F(x^{k+1}) > F(x^k) - c\|v(x^k)\|^2$ **then**
17:             $x^{k+1} = \mathtt{cutback}(x_k, x^{\mathrm{cg}}, d^k)$
18:             $k = k + 1$
19:             **break**
20:         **end if**
21:         $\rho^{k+1} = P(r^{k+1})$
22:         $d^{k+1} = -\rho^{k+1} + \frac{(r^{k+1})^T \rho^{k+1}}{(r^k)^T \rho^k} d^k$
23:         $k = k + 1$
24:     **end loop**
25: **end loop**

2.3. Convergence Analysis

We establish global convergence for algorithms iiCG-1 and iiCG-2 by showing that their constitutive steps, ISTA, subspace ISTA and CG, provide sufficient decrease in the

objective function. We also show a global 2-step Q-linear rate of convergence, and based on the fact that the number of cut CG steps cannot exceed half of the total number of steps, we establish a complexity result. In addition, we establish finite active set identification and termination for the two iiCG methods.

In this section, we assume that A is nonsingular, and denote its smallest and largest eigenvalues by λ and L, respectively. Thus, for any $x \in \mathbb{R}^n$,

$$\lambda \|x\|^2 \leq x^T A x \leq L \|x\|^2.$$

We denote the minimizer of F by x^*, and in the rest of this section we use the abbreviations

$$(2.1) \qquad v = v(x^k), \quad \omega = \omega(x^k), \quad \phi = \phi(x^k), \quad \psi = \psi(x^k), \quad g = g(x^k),$$

when convenient. We start by demonstrating a Q-linear decrease in the objective F for every ISTA step.

Lemma 2.3.1. *The ISTA step,*

$$x^{k+1} = \arg\min_y m^k(y) = x^k - \alpha\omega(x^k) - \alpha\psi(x^k),$$

with $0 < \alpha \leq 1/L$, satisfies

$$F(x^{k+1}) - F(x^*) \leq (1 - \lambda\alpha)(F(x^k) - F(x^*)).$$

Proof. Since $1/\alpha \geq L$, we have that $m^k(y)$ defined in (2.1) is a majorizing function for F at x_k. Therefore,

$$F(x^{k+1}) \leq m^k(x^{k+1}).$$

Since x^{k+1} is the minimizer of m^k, for any $d \in \mathbb{R}^n$, we have

$$F(x^{k+1}) \leq m^k(x^{k+1})$$
$$\leq m^k(x^k + \lambda\alpha d)$$
$$= F(x^k + \lambda\alpha d) - \tfrac{1}{2}(\lambda\alpha d)^T A(\lambda\alpha d) + \tfrac{1}{2\alpha}\|\lambda\alpha d\|^2$$

(2.2)
$$\leq F(x^k + \lambda\alpha d) + \tfrac{1}{2}\lambda^2\alpha(1 - \lambda\alpha)\|d\|^2.$$

Since F is a strongly convex function with parameter λ, it satisfies.

(2.3) $$\qquad F(tx + (1-t)y) \leq tF(x) + (1-t)F(y) - \tfrac{1}{2}\lambda t(1-t)\|x - y\|^2,$$

for any $x, y \in \mathbb{R}^n$ and $t \in [0, 1]$, see [**37**, Pages 63-64]. Setting $x \leftarrow x^k$, $y \leftarrow x^*$, and $t \leftarrow (1 - \lambda\alpha)$ (which is valid because $\lambda\alpha \in (0, 1]$), inequality (2.3) yields

(2.4) $$\quad F(x^k + \lambda\alpha(x^* - x^k)) \leq \lambda\alpha F(x^*) + (1 - \lambda\alpha)F(x^k) - \tfrac{1}{2}\lambda^2\alpha(1 - \lambda\alpha)\|x^* - x^k\|^2.$$

Since (2.2) holds for any d, we can set $d = x^* - x^k$. Substituting (2.4) in (2.2), we conclude that

$$F(x^{k+1}) - F(x^*) \leq (1 - \lambda\alpha)(F(x^k) - F(x^*)).$$

□

We now establish a similar result for the subspace ISTA step (2.1), under the conditions for which it is invoked, namely when the gradient balance condition is satisfied.

Lemma 2.3.2. *If $\|\omega(x^k)\| \leq \|\psi(x^k)\|$ the subspace ISTA step,*

$$(2.5) \qquad\qquad x^{k+1} = x^k - \alpha\psi(x^k),$$

with $0 < \alpha \leq 1/L$, satisfies

$$F(x^{k+1}) - F(x^*) \leq (1 - \tfrac{1}{2}\lambda\alpha)(F(x^k) - F(x^*)).$$

Proof. By the definitions (2.3) and (2.4), we have that $\omega^T\psi = 0$, and hence $\omega^T(x^{k+1} - x^k) = 0$ (Recall the abbreviations (2.1)). Given the pattern of zeros in x^k, ω and ψ, we have also that $\|x^{k+1} - \alpha\omega\|_1 = \|x^{k+1}\|_1 + \alpha\|\omega\|_1$. Using these two observations, we have for the full ISTA step,

$$m^k(x^k - \alpha\omega - \alpha\psi)$$

$$= f(x^k) + (-\alpha\omega - \alpha\psi)^T g + \frac{1}{2\alpha}\| -\alpha\omega - \alpha\psi\|^2 + \tau\|x^k - \alpha\omega - \alpha\psi\|_1$$

$$= f(x^k) + (x^{k+1} - \alpha\omega - x^k)^T g + \frac{1}{2\alpha}\|x^{k+1} - \alpha\omega - x^k\|^2 + \tau\|x^{k+1} - \alpha\omega\|_1$$

$$= f(x^k) + (x^{k+1} - x^k)^T g + \frac{1}{2\alpha}\|x^{k+1} - x^k\|^2 + \tau\|x^{k+1}\|_1 - \alpha\omega^T g + \frac{1}{2\alpha}\|\alpha\omega\|^2 + \tau\|\alpha\omega\|_1$$

$$= m^k(x^{k+1}) - \alpha\omega^T g + \frac{1}{2\alpha}\|\alpha\omega\|^2 + \tau\alpha\omega^T \operatorname{sgn}(\alpha\omega)$$

(2.6)
$$= m^k(x^{k+1}) - \frac{\alpha}{2}\|\omega\|^2,$$

where the last equality follows from the fact that, by (2.3), for all i s.t. $\omega_i \neq 0$, we have $\text{sgn}(g_i) = \text{sgn}(\omega_i)$, and that $w^T w = w^T g - \tau \omega^T \text{sgn}(g)$.

For the subspace ISTA step (2.5) we have

$$
\begin{aligned}
m^k(x^{k+1}) &= f(x^k) + (x^{k+1} - x^k)^T g + \frac{1}{2\alpha} \|x^{k+1} - x^k\|^2 + \tau \|x^{k+1}\|_1 \\
&= f(x^k) - \alpha \psi^T g + \frac{1}{2\alpha} \|\alpha \psi\|^2 + \tau \|x^{k+1}\|_1 \\
&= m^k(x^k) - \alpha \psi^T g + \frac{\alpha}{2} \|\psi\|^2 + \tau \|x^{k+1}\|_1 - \tau \|x^k\|_1.
\end{aligned}
$$

(2.7)

We examine the second term on the right hand side. For j such that $x_j^k = 0$ we have $\psi_j(x^k) = 0$. On the other hand, for j such that $x_j^k \neq 0$ definitions (2.2) and (2.1) yield

$$
x_j^{k+1} = x_j^k - \alpha \psi_j(x^k) = \arg\min_{y_j} f(x^k) + (y_j - x_j^k) g_j(x^k) + \tfrac{1}{2\alpha}(y_j - x_j^k)^2 + \tau |y_j|.
$$

In examining the optimality conditions of this problem there are two cases. If $x_j^{k+1} \neq 0$, we obtain $g_j = \psi_j - \tau \text{sgn}(x_j^{k+1})$. On the other hand, if $x_j^{k+1} = 0$ we have $g_j = \psi_j - \tau \sigma_j$

for some $\sigma_j \in [-1, 1]$. Substituting for g_j in (2.7), and using (2.7) gives

$$m^k(x^{k+1}) = m^k(x^k) + \sum_{j:x_j^{k+1} \neq 0} \tau\alpha\psi_j \operatorname{sgn}(x^{k+1})_j + \sum_{j:x_j^{k+1}=0} \tau\alpha\psi_j\sigma_j - \frac{\alpha}{2}\|\psi\|^2 + \tau\|x^{k+1}\|_1 - \tau\|x^k\|_1$$

$$= m^k(x^k) + \sum_{j:x_j^{k+1} \neq 0} \tau\alpha\psi_j \operatorname{sgn}(x^{k+1})_j + \sum_{j:x_j^{k+1}=0} \tau\alpha\psi_j\sigma_j - \frac{\alpha}{2}\|\psi\|^2$$

$$+ \tau(x^k - \alpha\psi)^T \operatorname{sgn}(x^{k+1}) - \tau\|x^k\|_1$$

$$= m^k(x^k) + \tau\sum_{j:x_j^{k+1} \neq 0} \left[x_j^k \operatorname{sgn}(x^{k+1})_j - |x_j^k|\right] + \tau\sum_{j:x_j^{k+1}=0} \left[x_j^k\sigma_j - |x_j^k|\right] - \frac{\alpha}{2}\|\psi\|^2$$

(2.8)

$$\leq m^k(x^k) - \frac{\alpha}{2}\|\psi\|^2.$$

Using in turn (2.6), the gradient balance condition $\|\psi\| \geq \|\omega\|$ and (2.8) we have

$$m^k(x^k) - m^k(x^k - \alpha\omega - \alpha\psi) = m^k(x^k) - m^k(x^{k+1}) + m^k(x^{k+1}) - m^k(x^k - \alpha\omega - \alpha\psi)$$

$$= m^k(x^k) - m^k(x^{k+1}) + \frac{\alpha}{2}\|\omega\|^2$$

$$\leq m^k(x^k) - m^k(x^{k+1}) + \frac{\alpha}{2}\|\psi\|^2$$

$$\leq 2[m^k(x^k) - m^k(x^{k+1})].$$

Since $m^k(x^k) = F(x^k)$, this relation yields

$$m^k(x^{k+1}) - F(x^k) \leq \tfrac{1}{2}[m^k(x^k - \alpha\omega - \alpha\psi) - F(x^k)].$$

Using this bound, the fact that m^k is a majorizing function of F, and that $(x^k - \alpha\omega - \alpha\psi)$ is a minimizer of m^k, we have that for any $d \in R^n$,

$$
\begin{aligned}
F(x^{k+1}) - F(x^k) &\leq \frac{1}{2}[m^k(x^k + \lambda\alpha d) - F(x^k)] \\
&= \frac{1}{2}[f(x^k) + g^T(\lambda\alpha d) + \frac{1}{2\alpha}\|\lambda\alpha d\|^2 + \tau\|x^k + \lambda\alpha d\|_1 - F(x^k)] \\
&= \frac{1}{2}[F(x^k + \lambda\alpha d) - F(x^k) - \tfrac{1}{2}(\lambda\alpha d)^T A(\lambda\alpha d) + \tfrac{1}{2\alpha}\|\lambda\alpha d\|^2] \\
&\leq \frac{1}{2}[F(x^k + \lambda\alpha d) - F(x^k) + \tfrac{1}{2}\lambda^2\alpha(1 - \lambda\alpha)\|d\|^2].
\end{aligned}
$$

(2.9)

Since F is a strongly convex function with parameter λ, it satisfies

(2.10) $$F(x + td) \leq F(x) + t(F(x + d) - F(x)) - \tfrac{1}{2}\lambda(1 - t)t\|d\|^2,$$

for any $x, d \in R^n$ and $t \in [0, 1]$. Setting $x \leftarrow x^k$, and $t \leftarrow \lambda\alpha$ (which is valid because $\lambda\alpha \in (0, 1]$), inequality (2.10) yields

$$F(x^k + \lambda\alpha d) \leq \lambda\alpha F(x^k) + (1 - \lambda\alpha)F(x^k + d) - \tfrac{1}{2}\lambda^2\alpha(1 - \lambda\alpha)\|d\|^2.$$

Substituting this inequality in (2.9), and setting $d = x^* - x^k$ we conclude that

$$
\begin{aligned}
F(x^{k+1}) - F(x^k) &\leq \tfrac{1}{2}(\lambda\alpha)(F(x^k + d) - F(x^k)) \\
&\leq \tfrac{1}{2}(\lambda\alpha)(F(x^*) - F(x^k)),
\end{aligned}
$$

which implies the result. $\qquad\square$

Next, we analyze the conjugate gradient step. To do so, we first introduce some notation and a technical lemma. The subgradient of $F(x)$ [37], of least norm, is given by

$$(2.11) \quad v_i(x) = \begin{cases} g_i(x) + \tau \operatorname{sgn}(x_i) & \text{if} \quad x_i \neq 0 \\ 0 & \text{if} \quad x_i = 0 \text{ and } |g_i(x)| \leq \tau \\ g_i(x) - \tau \operatorname{sgn}(g_i(x)) & \text{if} \quad x_i = 0 \text{ and } |g_i(x)| > \tau \end{cases} \quad \text{for} \quad i = 1, \ldots, n.$$

Recalling (2.3), we can write $v(x) = \omega(x) + \phi(x)$, where

$$(2.12) \qquad \phi_i(x) \triangleq \begin{cases} g_i(x) + \tau \operatorname{sgn}(x_i) & \text{if} \quad x_i \neq 0 \\ 0 & \text{if} \quad x_i = 0 \end{cases} \quad \text{for} \quad i = 1, \ldots, n.$$

The following result establishes a relationship between ψ and ϕ.

Lemma 2.3.3. *For all x^k, we have that $\|\psi(x^k)\| \leq \|\phi(x^k)\|$.*

Proof. We show that $|\psi_j| \leq |\phi_j|$ for each j such that $x_j \neq 0$ (the other components of the two vectors are zero). If $\phi_j = \psi_j$ the result holds, so assume $\phi_j \neq \psi_j$. For such j, we have that $x_j - \alpha \psi_j$ minimizes $m_j^k(y)$, which by (2.1) is given by

$$m_j^k(y) = f(x^k) + g_j(y - x_j^k) + \tfrac{1}{2\alpha}(y - x_j^k)^2 + |y|, \quad y \in \mathbb{R}.$$

On the other hand, $x_j - \alpha \phi_j$ minimizes

$$\bar{m}_j^k(y) \triangleq f(x^k) + g_j(y - x_j^k) + \tfrac{1}{2\alpha}(y - x_j^k)^2 + \operatorname{sgn}(x_j^k)y.$$

Thus $m_j^k(x_j^k - \alpha \psi_j) < m_j^k(x_j^k - \alpha \phi_j)$ and $\bar{m}_j^k(x_j^k - \alpha \psi_j) > \bar{m}_j^k(x_j^k - \alpha \phi_j)$, where the inequalities are strict since both functions are strictly convex. Subtracting these inequalities, we

have

$$m_j^k(x_j^k - \alpha\psi_j) - \bar{m}_j^k(x_j^k - \alpha\psi_j) < m_j^k(x_j^k - \alpha\phi_j) - \bar{m}_j^k(x^k - \alpha\phi).$$

Therefore,

(2.13) $\qquad 0 \leq |x_j^k - \alpha\psi_j| - \mathrm{sgn}(x_j^k)(x_j^k - \alpha\psi_j) < |x_j^k - \alpha\phi_j| - \mathrm{sgn}(x_j^k)(x_j^k - \alpha\phi_j),$

showing that the right hand side is positive, which implies that

(2.14) $\qquad\qquad\qquad\qquad \mathrm{sgn}(x_j^k) \neq \mathrm{sgn}(x_j^k - \alpha\phi_j).$

Now note that, since m_j^k and $\bar{m}_j^k$ coincide in a neighborhood of $x_j^k \neq 0$, ϕ_j and ψ_j have the same sign. We then consider two cases. If $\mathrm{sgn}(x_j^k) = \mathrm{sgn}(x_j^k - \alpha\psi_j)$, then by (2.14), the displacement $-\alpha\psi_j$ must be shorter than $-\alpha\phi_j$; therefore, $|\psi_j| < |\phi_j|$. On the other hand, if $\mathrm{sgn}(x_j^k) \neq \mathrm{sgn}(x_j^k - \alpha\psi_j)$, the left side of (2.13) is $2|x^k - \alpha\psi_j|$. By (2.14), the right side of (2.13) is $2|x_j^k - \alpha\phi_j|$. Thus $2|x_j^k - \alpha\psi_j| < 2|x_j^k - \alpha\phi_j|$. Since the displacement $-\alpha\phi_j$ produces a point farther from zero than the displacement $-\alpha\psi_j$, and since both displacements produce points with signs different from $\mathrm{sgn}(x_j^k)$, we have that $x_j^k - \alpha\phi_j$ is farther from x_j^k than $x_j^k - \alpha\psi_j$. Therefore, $|\psi_j| < |\phi_j|$.

$\square$

We can now show that a conjugate gradient step also guarantees sufficient decrease in the objective function, provided it is not truncated, i.e. that the `cutback` procedure is not invoked.

Lemma 2.3.4. *If Algorithms iiCG-1 or iiCG-2 take a full conjugate gradient step from x^k to x^{k+1}, then*

$$(2.15) \qquad F(x^{k+1}) \le F(x^k) - \beta \|v(x^k)\|^2,$$

where $\beta = \min\{c, 1/8L\}$, where c is defined in (2.9).

Proof. If $\operatorname{sgn}(x^{k+1}) \ne \operatorname{sgn}(x^{cg})$, then CG steps are accepted only if condition (2.9) is true, and hence (2.15) is satisfied. Otherwise $\operatorname{sgn}(x^{k+1}) = \operatorname{sgn}(x^{cg})$, and $F(x^{k+1}) = q(x^{k+1})$. Since we assume x^{k+1} is given by a full CG step, it follows that x^{k+1} is the result of a sequence of projected CG steps on problem (2.8), starting at x^{cg}. It is well known that a CG iterate x^{k+1} is a global minimizer of $q(\cdot\,; x^{cg})$ in the subspace $S = \operatorname{span}\{d^k, d^{k-1}, \ldots\}$, i.e.,

$$q(x^{k+1}) = \min\{q(x^k + y) : y \in S\}.$$

It is also known that $P(\nabla q(x^k; x^{cg})) \in S$, see [**39**, Theorem 5.3], and it follows from (2.12) that $P(\nabla q(x^k; x^{cg})) = \phi$. Thus,

$$F(x^{k+1}) = q(x^{k+1}) \le q(x^k - \zeta\phi),$$

for any ζ. Let us choose

$$\zeta = \frac{\phi^T \phi}{\phi^T A \phi}.$$

Recalling (2.6), we have

$$F(x^{k+1}) \leq \tfrac{1}{2}(x^k - \zeta\phi)^T A(x^k - \zeta\phi) + (-b + \tau \operatorname{sgn}(x^{\operatorname{cg}}))^T (x^k - \zeta\phi)$$

$$= F(x^k) + \frac{1}{2}\zeta\phi^T A\zeta\phi - \zeta\phi^T Ax^k + (-b + \tau \operatorname{sgn}(x^{\operatorname{cg}}))^T(-\zeta\phi)$$

$$= F(x^k) + \frac{\zeta}{2}\|\phi\|^2 - \zeta\phi^T(Ax^k - b + \tau \operatorname{sgn}(x^{\operatorname{cg}})).$$

By (2.12), for i such that $x_i^k = 0$ we have $\phi_i = 0$, and for $x_i^k \neq 0$ we have that $\phi_i = (Ax^k - b + \tau \operatorname{sgn}(x^{\operatorname{cg}}))_i$. Therefore,

$$F(x^{k+1}) = F(x^k) - \frac{\zeta}{2}\|\phi\|^2$$

$$\leq F(x^k) - \frac{1}{2L}\|\phi\|^2.$$

Since the step is only taken when condition (2.5) is true, we have from Lemma 2.3.3 that

$$\|v\| \leq \|\omega\| + \|\phi\| \leq \|\psi\| + \|\phi\| \leq 2\|\phi\|.$$

Therefore, we have that the following bound holds after one CG iteration,

$$F(x^{k+1}) \leq F(x^k) - \frac{1}{8L}\|v\|^2.$$

This implies (2.15) by definition of β. $\qquad\square$

We can now establish a 2-step Q-linear convergence result for both algorithms by combining the properties of their constitutive steps

Theorem 2.3.5. *Suppose that the stepsize α in the ISTA step (2.2) and the subspace ISTA step (2.1) satisfy $\frac{1}{8L} \leq \alpha \leq \frac{1}{L}$, and let $\beta = \min\{c, 1/8L\}$. Then, for the entire*

sequence $\{x^k\}$ generated by Algorithms iiCG-1 and iiCG-2 we have

$$(2.16) \qquad F(x^{k+2}) - F(x^*) \leq \left(1 - \frac{\lambda\beta}{2}\right)(F(x^k) - F(x^*)),$$

and thus $\{x^k\} \to x^*$.

Proof. By Lemma 2.3.1 and the lower bound on α, we have that the ISTA step satisfies

$$(2.17) \qquad F(x^{k+1}) - F(x^*) \leq (1 - \frac{\lambda}{16L})(F(x^k) - F(x^*)).$$

By Lemma 2.3.2, and the lower bound on α, we have that the subspace ISTA step also satisfies (2.17). By definition of β, relation (2.17) implies (2.16).

Now a that full CG steps provide the decrease (2.15). Convexity of F shows that

$$F(x^k) - F(x^*) \leq -v^T(x^* - x^k) \leq \|v\|\|x^* - x^k\|,$$

which combined with (2.15) gives

$$F(x^k) - F(x^{k+1}) \geq \beta \frac{(F(x^k) - F(x^*))^2}{\|x^* - x^k\|^2}.$$

Furthermore, since F is strongly convex, it satisfies

$$F(x^k) - F(x^*) \geq \tfrac{\lambda}{2}\|x^k - x^*\|^2,$$

see [**37**, pp. 63-64]. Using this bound we conclude that full CG steps satisfy (2.16).

Let us assume now that all CG steps terminated by the `cutback` procedure; i.e., that the worst case happens. After every such shortened CG step which may not provide sufficient reduction in F, the algorithm computes an ISTA step. Therefore, the Q-linear decrease (2.17) is guaranteed for every 2 steps, yielding (2.16) for all k.

Since the algorithms are descent methods, this implies the entire sequence satisfies $F(x^k) \to F(x^*)$ monotonically. Moreover, since F is strictly convex it follows that $x^k \to x^*$. $\qquad\square$

The most costly computations in our algorithms are matrix-vector products; the rest of the computations consist of vector operations. Therefore, when establishing bounds on the total amount of computation required to obtain an ϵ-accurate solution, it is appropriate to measure work in terms of matrix-vector products. Since there is a single matrix-vector product in each of the constitutive steps of our methods, a work complexity result can be derived from Theorem 2.3.5.

Corollary 2.3.6. *The number of matrix-vector products required by algorithms iiCG-1 and iiCG-2 to compute an iterate $\hat{x}$ such that*

$$(2.18) \qquad\qquad F(\hat{x}) - F(x^*) \leq \epsilon,$$

is at most

$$(2.19) \qquad\qquad \log\left[\frac{\epsilon}{F(x^0) - F(x^*)}\right] \Big/ \log\sqrt{1 - \frac{\lambda\beta}{2}}.$$

Proof. By (2.16), the condition (2.18), with $\hat{x} = x^{k+2}$, will be satisfied by an integer k such that

$$\left(1 - \tfrac{\lambda\beta}{2}\right)^{\frac{k}{2}} \left(F(x^0) - F(x^*)\right) \le \epsilon.$$

We obtain (2.19) by solving for k. $\qquad\qquad\square$

From the point of view of complexity, choosing $c = 1/8L$ is best, but in practice we have found it more effective to use a very small value of c since the strength of the conjugate gradient method is sometimes observed only after a few iterations are computed outside the initial orthant. More generally, Corollary 2.3.6 represents worst-case analysis, and is not indicative of the overall performance of the algorithm in practice. In our analysis we used the fact that a CG step is no worse than a standard gradient step – a statement that hides the power of the subspace procedure, which is evident in the finite termination result given next.

To prove that algorithms iiCG-1 and iiCG-2 identify the optimal active manifold and the optimal orthant in a finite number of iterations, we assume that strict complementarity holds. Since $v(x^*) = 0$, it follows from (2.11) that for all i such that $x_i^* = 0$ we must have $|g_i(x^*)| \le \tau$. We say that the solution x^* satisfies strict complementarity if $x_i^* = 0$ implies that $|g_i(x^*)| < \tau$.

Theorem 2.3.7. *If the solution x^* of problem (2.1) satisfies strict complementarity, then for all sufficiently large k, the iterates x^k will lie in the same orthant and active manifold as x^*. This implies that iiCG-1 and iiCG-2 identify the optimal solution x^* in a finite number of iterations.*

Proof. We start by defining the sets

$$Z^* = \{i : x_i^* = 0\}, \quad N^* = \{i : x_i^* < 0\}, \quad P^* = \{i : x_i^* > 0\},$$

and the constants

$$\delta_1 = \min_{i \in N^* \cup P^*} \frac{|x_i^*|}{2}, \quad \delta_2 = \min_{i \in Z^*} \left[\frac{\tau - |g_i(x^*)|}{2} \right].$$

Clearly $\delta_1 > 0$, and by the strict complementarity assumption we have that $\delta_2 > 0$.

Since, from Theorem 2.3.5 we have that $\{x^k\} \to x^*$, there exists an integer k_0 such that for any $k \geq k_0$ we have

$$x_i^k < -\delta_1 \ \forall i \in N^*, \quad x_i^k > \delta_1 \ \forall i \in P^*$$

(2.20)
$$|x_i^k| < \frac{\alpha \delta_2}{2} \ \forall i \in Z^*$$

(2.21)
$$|g_i| < \tau - \delta_2 \ \forall i \in Z^*.$$

For all such k, we have, by (2.21) that $\omega = 0$, which implies the behavior of iiCG-1 and iiCG-2 is identical.

Thus, all variables that are positive at the solution will be positive for $k > k_0$; and similarly for all negative variables. For the rest of the variables, we consider the ISTA step,

$$x^{k+1} = \max\{|x^k - \alpha g| - \alpha \tau, 0\} \operatorname{sgn}(x^k - \alpha g).$$

Using (2.20) and (2.21), we have that for any $k \geq k_0$ and $i \in Z^*$,

$$|x_i^k - \alpha g_i| - \alpha\tau \leq |x_i^k| + |\alpha g_i| - \alpha\tau$$
$$\leq \frac{\alpha\delta_2}{2} + \alpha(\tau - \delta_2) - \alpha\tau$$
$$= -\frac{\alpha\delta_2}{2} < 0.$$

Therefore, for all $i \in Z^*$ and all $k \geq k_0$, the ISTA step sets $x^{k+1} = 0$.

An ISTA step must be taken within n iterations of k_0, because of the finite termination property of the conjugate gradient algorithm. Therefore there exists a k_1 such that for any $k \geq k_1$, $\mathrm{sgn}(x^k) = \mathrm{sgn}(x^*)$, and by (2.21), $\omega = 0$. These two facts imply that for $k \geq k_1$, once the algorithm enters the CG iteration it will not leave, since the two `break` conditions cannot be satisfied. Finite termination of CG implies the optimal solution x^* will be found in a finite number of iterations. □

2.4. Numerical Results

We developed a MATLAB implementation[1] of algorithms iiCG-1 and iiCG-2, and in the next subsection we compare their performance relative to two well known proximal gradient methods. This allows us to study the algorithmic components of our methods in a controlled setting, and to identify their most promising characteristics. Then we compare, in Section 2.4.3, our algorithms to four state-of-the-art codes for solving problem (2.1). Our numerical experiments are performed on four groups of test problems with varying characteristics.

---

[1]Available at `https://github.com/stefanks/Ql1-Algorithm`

Before describing the numerical tests, we introduce the following heuristic that im-proves the prediction made by ISTA steps. As suggested by Wright et al. [**54**], the Barzilai-Borwein stepsize with a non-monotone line-search is usually preferable to a con-stant stepsize scheme, such as α in algorithms iiCG-1 and iiCG-2. Thus Line 3 in iiCG-1 and Lines 4 and 6 in iiCG-2 are replaced by the following procedure, where we now write ψ in the form $\psi(x^k; \alpha_B)$ to make its dependence on the steplength α_B clear.

ISTA-BB-LS Step

1: $\alpha_B = \frac{(x^k - x^{k-1})^T (x^k - x^{k-1})}{(x^k - x^{k-1})^T A(x^k - x^{k-1})}$

2: **repeat**

3:    $x_F = x^k - \alpha_B \omega(x^k) - \alpha_B \psi(x^k; \alpha_B)$ (when the ISTA step is invoked)

4:    or

5: $x_F = x^k - \alpha_B \psi(x^k; \alpha_B)$ (when the reduced ISTA step is invoked)

6:    $\alpha_B = \frac{\alpha_B}{2}$

7: **until** $F(x_F) \leq \max_{i \in \{1...M\}} F^i - \alpha_B \xi \|x - x_F\|^2$

8: $F^{i+1} = F^i$ for all $i \in \{1 \ldots M - 1\}$

9: $F^1 = F(x_F)$

10: $x^{k+1} = x_F$

At the beginning of the overall algorithm, we initialize $M = 5$, $\xi = 0.005$, and $F^i = F(x^0)$ for $i \in \{1 \ldots M\}$. The choice of parameters M, ξ is as in [**54**]; we did not attempt to fine tune them to our test set.

2.4.1. Initial Evaluation of the Two New Algorithms

We implemented the following methods.

iiCG-1

iiCG-2

FISTA The Fast Iterative Shrinkage-Thresholding Algorithm [**3**], using a constant stepsize given by $1/L$.

ISTA-BB-LS This method is composed purely of the ISTA-BB-LS steps described at the beginning of this section, which are repeated until convergence.

These four methods allow us to perform a per-iteration comparison of the progress achieved by each method. FISTA and ISTA-BB-LS are known to be efficient in practice and serve as a useful benchmark. In algorithms iiCG-1 and iiCG-2 we set $c = 10^{-4}$, and set $\alpha = 1/L$ in (2.4) when computing the value of $\psi(x^k)$ used in the gradient balance condition (2.5). As illustrated in Appendix .2, our algorithms are fairly insensitive to the choice of this parameter.

The first three test problems have the following form, which is sometimes called the elastic net regularized least squares problem [**55**],

$$(2.1) \qquad \min_x \tfrac{1}{2}\|y - Bx\|^2 + \gamma\|x\|^2 + \tau\|x\|_1.$$

The data y and B was obtained from three different data sets that we call `spectra`, `sigrec`, and `randQuad`. The sources of these data sets are as follows.

Spectra. The gasoline spectra problem is a regularized linear regression problem [**31**]; it is available in MATLAB by typing `load spectra`. This problem has a slightly different form than (2.1) in the sense that ℓ_1 regularization is imposed on all but one of the variables (which represents the constant term in linear regression).

Sigrec. This signal recovery problem is described by Wright et al. [**54**]. The authors generate random sparse signals, introduce noise, and encode the signals in a lower dimensional vector by applying a random matrix multiplication. We generated an instance using the code by the authors of [**54**].

Myrand. We generated a random 2000 variable problem using the following MATLAB commands

```
B = randn(1000,2000); y = 2000*randn(1000,1),
```

and employed this matrix and vector in (2.1).

The 4th problem in our test set is of the form

(2.2) $$\min_x \tfrac{1}{2} x^T C x - y^T x + \gamma \|x\|^2 + \tau \|x\|_1.$$

Proxnewt. The data for (2.2) was generated by applying the proximal Newton method described in [**7**] to an ℓ_1 regularized convex optimization problem of the form $\varphi(x) + \tau\|x\|_1$, where $\varphi(x)$ is a cross entropy loss function and the data is given by the `gisette` test set in the LIBSVM repository [**10**]. Each iteration of the proximal Newton method computes a step by solving a subproblem of the form (2.1). We extracted one of these subproblems, and added the ℓ_2 regularization term $\gamma\|x\|^2$ to yield a problem of the form (2.2).

We created twelve versions of each of the four problems listed above, by choosing different values of γ and τ. This allowed us to create problems with various degrees of ill conditioning and different percentages on non-zeros in the solution. In our datasets, the matrices B and C in (2.1) and (2.2) are always rank deficient; therefore, when $\gamma = 0$ the Hessians of $f(x)$ are singular.

The following naming conventions are used. The last digit, as in problems

$$\text{spectras}\mathbf{1}, \cdots, \text{spectras}\mathbf{4},$$

indicates one of the four values of τ that were chosen for each problem so as to generate different percentages of non-zeros in the optimal solution. The degree of ill conditioning, which is controlled by γ, is indicated in the second-to-last character, as in

$$\text{spectra}\mathbf{s}1, \quad \text{spectra}\mathbf{i}1, \quad \text{spectra}\mathbf{m}1,$$

which correspond to the **s**ingular, **i**ll conditioned and **m**oderately conditioned versions of the problem. The characteristics of the test problems are given in Appendix .1.

Accuracy in the solution is measured by the ratio

$$(2.3) \qquad\qquad \texttt{tol} = \frac{F(x^k) - F^*}{|F^*|},$$

where F^* is the best known objective value for each problem. Given the nature of the four algorithms listed above, it is easy to compute and report the ratio (2.3) after each matrix-vector product computation. We initialized x^0 to the zero vector, and imposed a limit of 50,000 matrix-vector products on all the runs.

In Table 2.1 we report the results for iiCG-1, iiCG-2, FISTA and ISTA-BB-LS on all the test problems. Matrix-Vector product (MV) counts are a reasonable measure of computational work used by these algorithms since they are by far the costliest operations. All algorithms are terminated as soon as the ratio in (2.3) is less than a prescribed constant; in Table 2.1 we report results for $\texttt{tol} = 10^{-4}$, and $\texttt{tol} = 10^{-10}$. Dashes signify failures to find a solution after 50,000 matrix-vector products, and bold numbers mark the best-performing algorithm.

We observe from these tables that problems with an intermediate value of τ (typically) require the largest effort. This suggests that the values of τ chosen in our tests gave rise to an interesting collection of problems that range from nearly quadratic to highly regularized piecewise quadratic, with the most challenging problems in the middle range. An analysis of the data given in Table 2.1 indicates that iiCG-2 is the most efficient in these tests, but not uniformly so. Overall, we regard both iiCG-1 and iiCG-2 as promising methods for solving the regularized quadratic problem (2.1).

Table 2.1. Number of matrix-vector products to reach accuracies `tol` $= 10^{-4}$ and $= 10^{-10}$.

	$\mathtt{tol} = 10^{-4}$				$\mathtt{tol} = 10^{-10}$			
	iiCG-1	iiCG-2	FISTA	ISTA-BB-LS	iiCG-1	iiCG-2	FISTA	ISTA-BB-LS
randQuads1	654	297	**248**	1311	13614	**8102**	8511	-
randQuads2	513	310	**163**	547	3228	**1885**	4748	12782
randQuads3	118	123	**69**	86	398	**311**	1493	540
randQuads4	11	12	21	**8**	18	20	106	**17**
randQuadi1	**14**	**14**	31	19	-	-	**26183**	-
randQuadi2	491	297	**163**	498	3008	**1912**	4925	13682
randQuadi3	111	116	**69**	86	352	**335**	1492	596
randQuadi4	11	12	21	**8**	18	20	106	**17**
randQuadm1	**14**	**14**	30	19	**57**	**57**	236	726
randQuadm2	121	**108**	111	317	1731	**728**	2437	5483
randQuadm3	117	128	**68**	86	375	**359**	1433	466
randQuadm4	11	12	21	**8**	18	19	106	**17**
spectras1	**4**	**4**	265	17	-	**45888**	-	-
spectras2	**4**	**4**	264	20	48200	**8656**	-	-
spectras3	**4**	**4**	263	26	5661	**2245**	-	-
spectras4	**4**	**4**	270	22	30896	**9170**	-	-
spectrai1	**4**	**4**	258	23	**42**	**42**	29258	12046
spectrai2	**4**	**4**	257	26	159	**129**	33734	-
spectrai3	**4**	**4**	256	19	2246	**2205**	45339	-
spectrai4	**60**	105	1036	4192	1898	**1751**	10030	23579
spectram1	**2**	**2**	**2**	**2**	**10**	**10**	1897	17
spectram2	**2**	**2**	**2**	**2**	15	**12**	2024	137
spectram3	**5**	**5**	51	7	**11**	**11**	1445	163
spectram4	**100**	**100**	126	175	**107**	**107**	4799	545
sigrecs1	2206	1213	**446**	3522	3635	2283	**1338**	6687
sigrecs2	1020	589	**291**	1494	1191	**695**	696	1737
sigrecs3	105	94	75	**72**	120	116	296	**86**
sigrecs4	11	12	25	**8**	19	20	145	**16**
sigreci1	8	8	14	10	-	5415	10542	-
sigreci2	2148	1156	**442**	3515	3526	2190	**1350**	6955
sigreci3	1095	532	**291**	1499	1285	**637**	659	1728
sigreci4	11	12	25	**8**	19	20	145	**16**
sigrecm1	**8**	**8**	14	10	**51**	**51**	114	386
sigrecm2	65	**60**	61	101	777	**297**	864	1138
sigrecm3	199	137	**103**	229	476	**365**	1301	504
sigrecm4	11	12	25	**8**	18	19	144	**16**
proxnewts1	22739	**4077**	21173	-	40139	**10169**	-	-
proxnewts2	9522	6871	**6423**	38233	15782	**8436**	-	-
proxnewts3	6463	**1865**	2026	3659	7092	**2098**	-	8384
proxnewts4	212	**191**	1582	311	233	**213**	-	458
proxnewti1	294	**336**	2795	49932	3267	**1100**	-	-
proxnewti2	2274	**1384**	2212	14029	3519	**1983**	-	24756
proxnewti3	6076	**1437**	1702	3027	6585	**1647**	-	6344
proxnewti4	291	**160**	1499	296	315	**175**	-	463
proxnewtm1	**32**	**32**	881	190	131	**112**	20319	2382
proxnewtm2	41	**36**	784	293	139	**101**	14310	1369
proxnewtm3	237	**232**	592	492	262	274	12640	720
proxnewtm4	58	**50**	472	101	70	**58**	10380	128

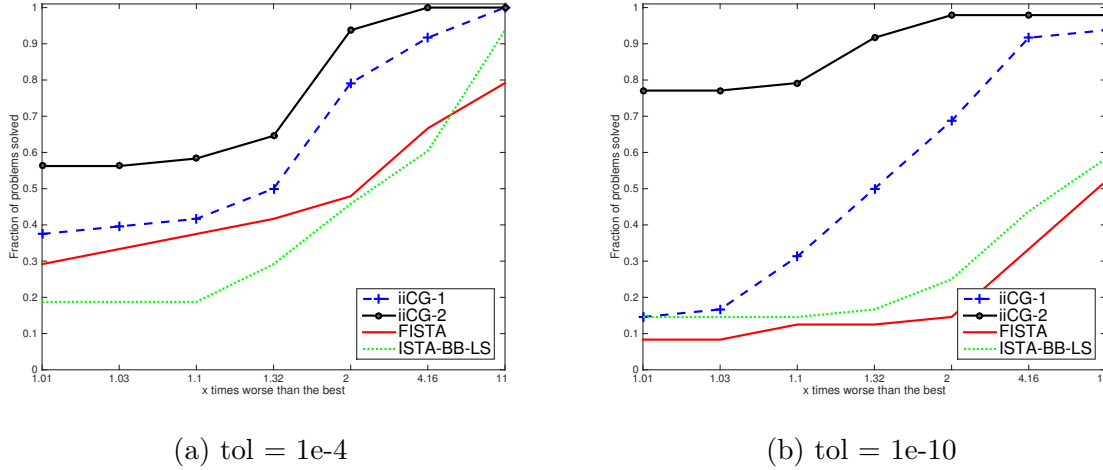(a) tol = 1e-4   (b) tol = 1e-10

Figure 2.1. Comparison of the four algorithms using the logarithmic Dolan-Moré profiles, based on the number of matrix-vector products required for convergence. We report results for low and high accuracy (2.3) in the objective function.

Using the data from Table 2.1, we illustrate in Figure 2.1 the relative performance of iiCG-1, iiCG-2, FISTA and ISTA-BB-LS, using the Dolan-Moré profiles [**15**] (based on the number of matrix-vector multiplications required for convergence). While both iiCG-1 and iiCG-2 are efficient in the case `tol` $= 10^{-4}$, iiCG-2 demonstrates superior performance in reaching the higher accuracy.

In Figure 2.2 we illustrate typical behavior of iiCG-1 and iiCG-2 by means of problems `proxnewts3` and `spectram4`. We plot the accuracy in the objective (2.3) as a function of of matrix-vector multiplications. Both plots show that our algorithms are able to estimate the solution to high accuracy. They sometimes outperform the other methods from the very start, as in Figure 2.2a, but in other cases iiCG-1 and iiCG-2 show their strength later on in the runs; see Figure 2.2b. We note that the ISTA-BB-LS method is more efficient than FISTA when high accuracy in the solution is required.
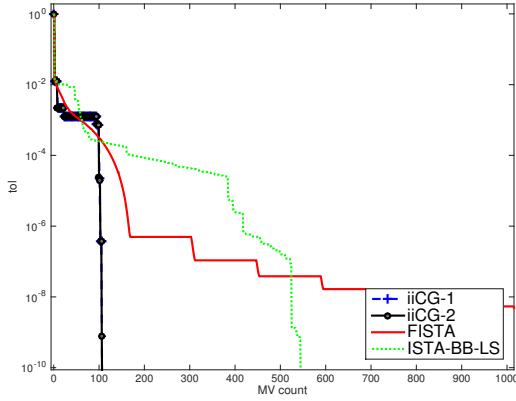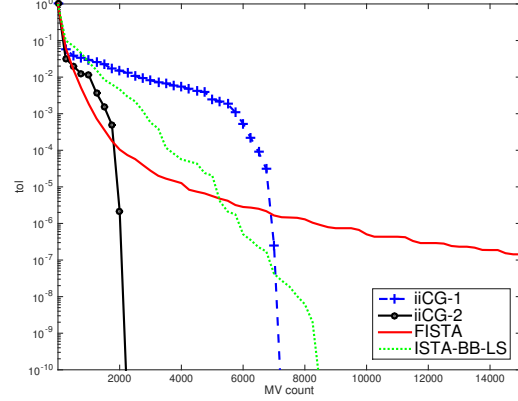
(a) Problem `spectram4`

(b) Problem `proxnewts3`

Figure 2.2. Accuracy (2.3) in the objective function (vertical axis) as a function of the number of matrix-vector products performed (MV count) performed by each algorithm

We have observed that Algorithm iiCG-2 is superior to iiCG-1 in identifying sparse solutions, due to its judicious application of the subspace ISTA iteration. To illustrate this, we plot in Figure 2.3 the Pareto frontier based on two quantities: the accuracy (2.3) in the solution, and the number of nonzero elements in the solution. Specifically, we ran the test problems, `spectras2` and `randQuads2` until a specified limit of matrix-vector (MV) products was computed (500, 2100, 30000 for `spectras2` and 500, 1000, 2000 for `randQuads2`). For each run, all iterates were considered, and we plotted the pairs (accuracy-nonzeros) such that no other pair existed with both higher accuracy and sparsity. As expected, when more effort (MV) is allowed, higher accuracy is achieved, but not necessarily higher sparsity. As measured by the two quantities depicted in Figure 2.3, iiCG-2 finds better solutions.
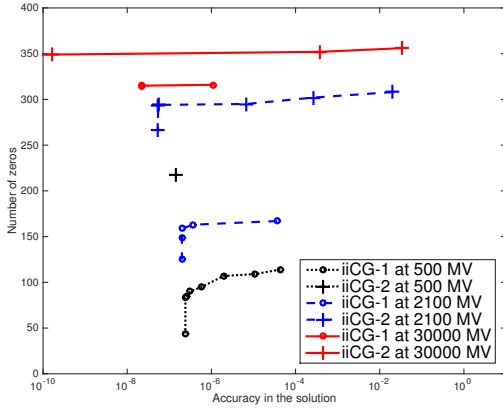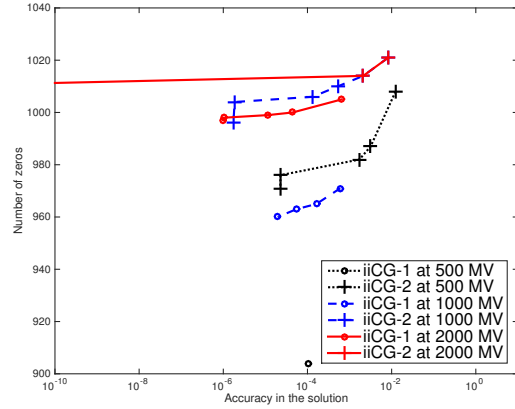
(a) On problem `spectras2` (b) On problem `randQuads2`

Figure 2.3. Pareto frontier based on the number of nonzeros in the incumbent solution (vertical axis) and accuracy in the solution (2.3) (horizontal axis), for runs imposing 3 limits on the maximum number of matrix-vector (MV) products. The test problems, `spectras` and `randQuads2`, represent typical behavior of the algorithms.

2.4.2. Analysis of the CG Phase

We now discuss the behavior of the subspace CG phase, which has a great impact on the overall efficiency of the proposed algorithms. In Figure 2.4 we report data for two representative runs of iiCG-2 given by test problems `randQuadm1` and `sigreci4`. The horizontal axis labels each of the subspace phases invoked by the algorithm, and the vertical axis gives the number of CG iterations performed during that subspace phase.

Figure 2.4a illustrates a behavior that is often observed for moderate and large values of the penalty parameter τ, namely that the bulk of the CG iterations are performed towards the end of the run. This is desirable, as the CG phase makes a moderate contribution earlier on towards identifying the optimal active set, and is then able to compute a highly accurate solution of the problem in one (or two) CG cycles. Figure 2.4b, considers the case

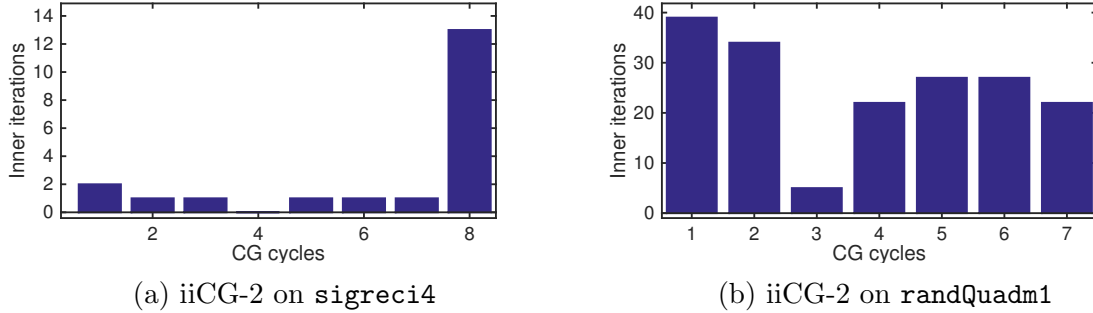(a) iiCG-2 on `sigreci4`   (b) iiCG-2 on `randQuadm1`

Figure 2.4. Number of CG iterations in each subspace phase

when the penalty parameter τ is very small, i.e., when F is nearly quadratic. We observe now that the effort expended by the CG phase is more evenly distributed throughout the run. It is reassuring that the number of CG iterations does not tail off for this problem, and that a significant number of CG steps is performed in the last cycle, yielding an accurate solution to the problem.

2.4.3. Comparisons with Established Codes

We also performed comparisons with the following four state-of-the-art codes. To facilitate our comparisons, and ease of implementation, we only considered codes written in MATLAB.

- **SPARSA** This is the well known implementation of the ISTA method described in [54]. The code can be found at `http://www.lx.it.pt/~mtf/SpaRSA/`
- **PSSgb** The algorithm implemented in this code is motivated by the two-metric projection method [26] for bound constrained optimization. That method is extended to the regularized ℓ_1 problem; curvature information is incorporated

in the form of a BFGS matrix. `http://www.di.ens.fr/~mschmidt/Software/thesis.html`

- **N83** Is one of the codes provided by the TFOCS package [**4**]. It implements the optimal first order Nesterov method described in [**36**]. `http://cvxr.com/tfocs/download/`

- **pdNCG** A Newton-CG method in which the ℓ_1 norm is approximated by a smooth function [**21**]. `http://www.maths.ed.ac.uk/~kfount/index.html`

We also experimented with SALSA [**1**], TWIST [**6**] and FPC_AS [**53**], l1_ls [**33**], YALL1 [**14**], but these codes were not competitive on our test set with the four packages mentioned above.

In Figure 2.5 we compare our algorithms with the four codes listed above on all test problems. The figure plots the Dolan-Moré performance profiles based on CPU time; we report results for two values of the convergence tolerance (2.3). Figure 2.5 indicates that PSSgb is the closest in performance compared to our algorithms.

We now examine the behavior of the codes for intermediate values of accuracy. Figure 2.6 shows the fraction of problems that a method was able to solve faster than the other methods, as a function of the accuracy measure (2.3), in a log-scale.

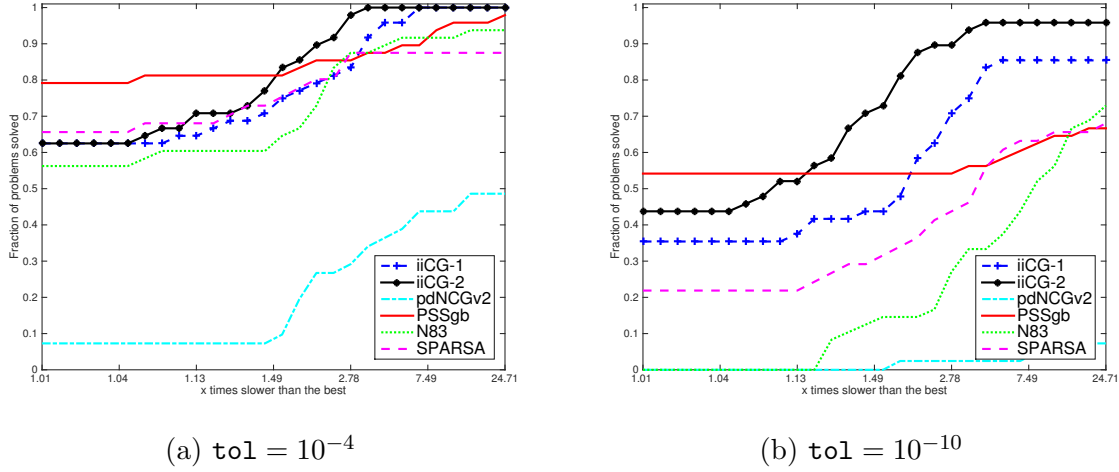(a) $\texttt{tol} = 10^{-4}$ (b) $\texttt{tol} = 10^{-10}$

Figure 2.5. Comparison of iiCG-1, iiCG-2, SPARSA, N83 and PSSgb. The figure plots the logarithmic Dolan-Moré performance profiles based on CPU time for problems `spectra`, `sigrec` and `randQuad`.
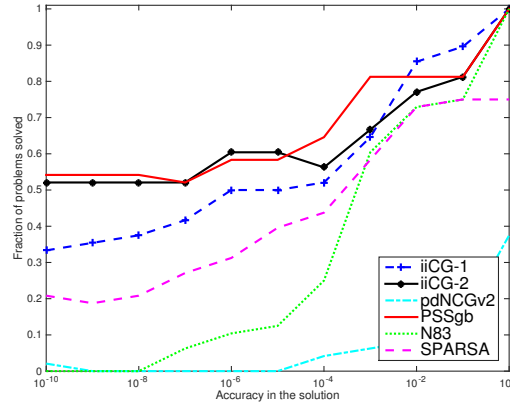

Figure 2.6. Efficiency. For given accuracy in the function value (horizontal axis), the plot shows the percentage of problems solved to that accuracy within 10% of the time of the best method.

Some algorithms are better suited for problems with a very high solution sparsity. In Figure 2.7, we show the Dolan-Moré profiles for two sets of problems; one with low sparsity in the solution, and one with high sparsity. The first set of test problems is composed of problems styled *name1* (average solution sparsity 25%), and the second is

made of problems styled *name4* (average solution sparsity 95%). We observe that iiCG-2 outperforms the other codes for low solution sparsity, and that PSSgb is competitive for high sparsity problems. We did not include pdNCGv2 and SPARSA since they are not competitive with the other algorithms, in these tests.

Overall, our iiCG methods are competitive with the four state-of-the-art codes in these tests.
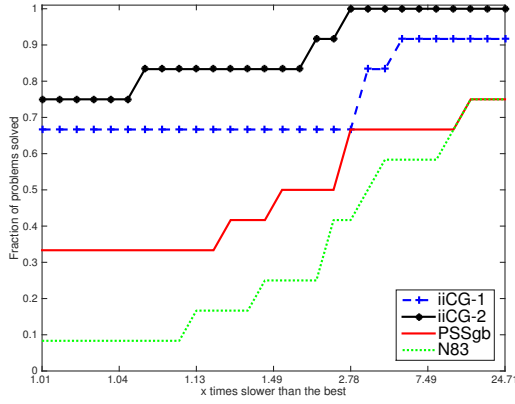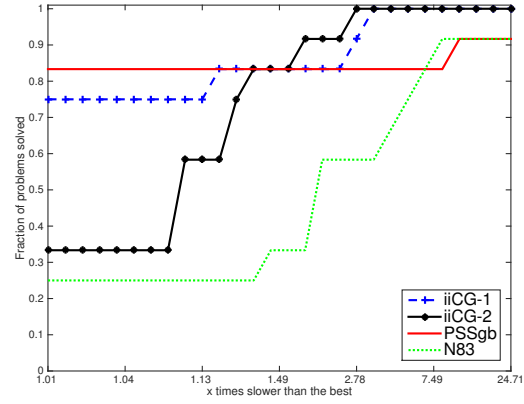
(a) Low Sparsity   (b) High Sparsity

Figure 2.7. Comparison of four algorithms using the logarithmic Dolan-Moré profiles, for problems with: low sparsity (Figure a), and high sparsity (Figure b). We set $tol = 10^{-7}$

2.5. Final Remarks

In this chapter, we presented a second-order method for solving the ℓ_1 regularized quadratic problem (2.1). We call the method iiCG, for "interleaved ISTA-CG" method. It differs from other second-order methods proposed in the literature in that it can invoke one of two possible steps at each iteration: a subspace conjugate gradient step or a first order active-set identification step. This flexibility is designed to allow the algorithm to adapt itself to the problem to be solved, and the results presented in the chapter suggest that it is generally successful at achieving this goal. The decision of what type of step to invoke is based on a measure related to the relative components of the minimum norm subgradient — an idea proposed by Dostal and Schoeberl [17] for the solution of bound constrained quadratic optimization problems. But unlike [17], our algorithm does not include a step along the direction $-\omega(x^k)$ in order to relax zero variables; as a result our approach departs significantly from the methodology given in [17].

We presented two variants of our method that differ in the first-order active set identification phase. One option uses ISTA, and the other a subspace ISTA iteration.

To provide a theoretical foundation for our method, we established global rates of convergence and complexity bounds based on the total amount of work expended by the algorithm (measured by the total number of matrix-vector products performed). We also gave careful consideration to the main design components of the algorithm, such as the definition of the vectors ω, ψ employed in the gradient balance condition (2.5). One of the features of the algorithm that was particularly successful is allowing the CG iteration to cross orthants as long as sufficient decrease in the objective function is achieved. The

numerical tests reported suggest that the algorithm proposed in this chapter is competitive with state-of-the-art codes.

CHAPTER 3

An Evolving Gradient Resampling Method

The stochastic gradient method of Robbins-Monro [**44**] is the algorithm of choice for optimization problems arising in many large-scale machine learning applications. At the onset of the optimization procedure, while the number of processed datapoints is relatively small, it often outperforms most other methods proposed in the literature. However, to achieve such efficient initial behavior, the steplength must be chosen to be sufficiently large. This prevents the algorithm from converging on a good solution later on. This is due to the high variance of the stochastic gradients, which must be controlled with a diminishing steplength. Several algorithms have been proposed to address this limitation, including methods that average the iterates [**43, 46, 38**], dynamic sampling methods [**9, 22, 41**] and aggregated gradient methods [**45, 30, 49, 34, 13, 24, 12**]. The latter focus on the finite sum problem and enjoy a linear rate of convergence for strongly convex problems, whereas the stochastic gradient method (in this case, the incremental gradient method) only has a sublinear rate of convergence.

In this chapter, we propose an algorithm that shares some characteristics with the aggregated gradient methods mentioned above, but is more general and flexible. Unlike those methods, which are designed to revisit the training set multiple times (perhaps dozens of times), the algorithm proposed here is well suited for problems involving an extremely large training set, and where one (or a few) passes over the data suffice to produce an acceptable solution. The method is quite general; it includes as special cases

the classical stochastic gradient method, the aggregated gradient methods SAG [12] and SAGA [12] and dynamic sampling methods [9, 22, 41]. The ultimate goal of this work is the design of an algorithm that achieves good testing error—not just good training error– a goal it shares with the streaming SVRG method recently described in [24].

The problem of interest is stated as

$$\text{(3.1)} \qquad \min_{x \in \mathbb{R}^n} F(x) = \mathbb{E}[f(x; \xi)],$$

where ξ is a random variable with distribution P and $f(\cdot\,; \xi) : \mathbb{R}^n \to \mathbb{R}$ is a smooth function. In machine learning applications, f is the composition of a loss function and a prediction function parametrized by a vector x. For purposes of analysis, we assume that $f(\cdot\,; \xi)$ is strongly convex, but the algorithm is well defined and practical even for non-convex problems.

Iterations of the stochastic gradient (SG) method for solving problem (3.1) are given by

$$\text{(3.2)} \qquad x_{k+1} = x_k - \alpha_k \nabla f(x_k; \xi_k),$$

where α_k is a steplength and ξ_k is a realization of the random variable ξ. Iterations of the method proposed in this project have the more general form

$$\text{(3.3)} \qquad x_{k+1} = x_k - \alpha y_k,$$

where y_k is computed as a weighted average of the sample gradients $\nabla f(x_i, \xi_j)$ evaluated at previous iterations and sample gradients $\nabla f(x_k, \xi_j)$ evaluated at the current iterate

x_k (Here ξ_j simply stand for some past realizations of the random variable ξ). The algorithm is designed so that the amount of gradient information contained in y_k increases monotonically during the course of the optimization procedure. This improves the quality of the vector y_k compared to the aggregated gradient methods described below and allows the algorithm to achieve a low value of the objective F. We refer to the algorithm as the Evolving Gradient Resampling Method, or EGR in short. We show that, for strongly convex objective functions, the expected value of $F(x_k)$ converges to the optimal value F^* at a linear rate and that the computational complexity bounds compare well with those of established methods.

This chapter is organized into 6 sections. In the next section we present the algorithm, and in section 3.2 we study its convergence properties. A detailed discussion of the implementation of the algorithm is given in section 3.3 and the results of numerical experiments are reported in section 3.4. The contributions of the work and its relationship to other works in the literature are summarized in section 3.5.

Notation and Definitions. We denote the minimizer of F by x^*, and the corresponding objective value by F^*. Following Bertsekas [5] we use the term "incremental gradient method" for the method designed to minimize a finite sum problem and "stochastic gradient" for the same method for minimizing empirical risk F.

3.1. The Evolving Gradient Resampling (EGR) Method

The EGR method combines ideas from dynamic sampling methods, which employ mini-batches of increasing size, and aggregated gradient methods that store gradients evaluated at previous iterations. The latter, which are more accurately described as

randomized incremental aggregated methods [**45, 30, 24, 12**], employ an iteration of the form (3.3) to solve the finite sum problem

$$(3.4) \qquad \frac{1}{m} \sum_{i=1}^{m} f^i(x),$$

where $f^i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ are given convex functions. An example of a method of this kind is SAG [**45**], where at every iteration k an index $j \in \{1, \ldots, m\}$ is chosen at random and the aggregated gradient y_k is computed as

$$(3.5) \qquad y_k^{sag} = \frac{1}{m} \left[\nabla f^j(x_k) - \nabla f^j(\phi_{k-1}^j) + \sum_{i=1}^{m} \nabla f^i(\phi_{k-1}^i) \right].$$

Here ϕ_{k-1}^i is the most recent iterate at which the gradient of the function f^i was evaluated. Thus, in the SAG method one gradient, namely ∇f^j, is updated while the rest of the gradients kept in storage (i.e. those in the summation in (3.5)) are not altered. A distinctive property of SAG and the methods described in [**30, 12, 49, 34, 13**] is that they achieve a linear rate of convergence in the expected value of the finite sum problem, which is not the case for the incremental gradient method. Numerical experience has shown that the SAG, SAGA and SVRG[**30**] methods are effective in practice.

Aggregated gradient methods are suitable for the case when m is not too large so that multiple passes over the data are possible. Storage is a concern, as m gradients need to be saved, which may not be practical in general for applications involving large data sets. However, when f^i is the composition of a linear function and a smooth function (as in logistic regression) it suffices to store one scalar in lieu of the gradient vector [**45**]. Storage

is not an issue for the SVGR method [**30**], which requires only one extra vector of storage regardless of the form of f^i.

The EGR method maintains an integer t_k that counts the number of gradient vectors in storage at the beginning of the k-the iteration, and defines the set $T_k = \{1, \ldots, t_k\}$. The objective function F can then be approximated by the sample average approximation

$$(3.6) \qquad F_{T_k}(x) = \frac{1}{t_k} \sum_{i=1}^{t_k} f(x; \xi_i) \equiv \frac{1}{t_k} \sum_{i=1}^{t_k} f^i(x),$$

where we have defined

$$(3.7) \qquad f^i(x) = f(x, \xi_i),$$

so that F_{T_k} is similar in form to the (deterministic) finite sum function (3.4). Iterations of the EGR method are given by

$$(3.8) \qquad x_{k+1} = x_k - \alpha y_k,$$

where α is a (fixed) steplength and the aggregated gradient y_k is defined as follows: At the k-th iteration, the method adds some new sample gradients, indexed by the set U_k, to y_k, and at the same time updates some of the gradients previously stored, indexed by S_k. More precisely, defining

$$(3.9) \qquad u_k = |U_k|, \qquad s_k = |S_k|,$$

the first version of our method sets

$$(3.10) \quad y_k = \frac{1}{t_k + u_k} \left(\sum_{j \in S_k} [\nabla f^j(x_k) - \nabla f^j(\phi^j_{k-1})] + \sum_{i=1}^{t_k} \nabla f^i(\phi^i_{k-1}) + \sum_{j \in U_k} \nabla f^j(x_k) \right),$$

where ϕ^j_{k-1} denotes the most recent iterate at which the gradient of function the f^j was computed (we define this precisely below).

Formula (3.10) is motivated by the SAG method (3.5). The main changes are that the second sum in (3.10) includes t_k terms rather than m terms, that the first sum accounts for the fact that s_k gradients in storage are updated, and the last sum represents the addition of u_k gradients evaluated at the current iterate. In the EGR method where $t_{k+1} = t_k + u_k$, we have the option of having $u_k > 0$, so that the number of gradients contained in y_k grows at every iteration, in contrast with methods described in [**30, 12, 49, 34, 13**], where the number of stored gradients is constant. (The SAG method corresponds to the choice $s_k = 1, u_k = 0, t_k = m$).

An alternative version EGR is based on the SAGA step direction,

$$(3.11) \quad y_k^{saga} = \nabla f^j(x_k) - \nabla f^j(\phi^j_{k-1}) + \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\phi^i_{k-1}),$$

and also allows for multiple sample gradient replacements as well as addition of new samples to memory. The step direction is then defined as

(3.12)

$$y_k = \frac{1}{s_k + u_k} \left(\sum_{j \in S_k} \left[\nabla f^j(x_k) - \nabla f^j(\phi^j_{k-1}) + \frac{1}{t_k} \sum_{i=1}^{t_k} \nabla f^i(\phi^i_{k-1}) \right] + \sum_{j \in U_k} \nabla f^j(x_k) \right).$$

where, as before, ϕ_{k-1}^j denotes the most recent iterate at which the gradient of the function f^j was computed. A pseudocode of the algorithm, which can employ either definition of y_k is given in Algorithm 3

Algorithm 3 EGR Algorithm

Input: Sequences $\{s_k\}$, $\{u_k\}$, steplength α, and initial iterate x_0

1: Set $t_0 = 0$

2: **loop** $k = 0, 1, 2, , \ldots$

3:   $U_k = \{t_k + 1, \ldots, t_k + u_k\}$

4:   For all $j \in U_k$ draw ξ_j from the distribution P

5:   $S_k =$ sample of s_k indices from $\{1, \ldots, t_k\}$

6:   Evaluate $\nabla f^j(x_k) = \nabla f(x_k, \xi_j)$ for $j \in S_k \cup U_k$

7:   Compute y_k by (3.10) or (3.12)

8:

$$(3.13) \ \text{Set} \ \phi_k^i = \begin{cases} x_k & \text{for all } i \in S_k \cup U_k \\ \phi_{k-1}^i & \text{otherwise} \end{cases}$$

9:   $x_{k+1} = x_k - \alpha y_k$

10:   $t_{k+1} = t_k + u_k$

11: **end loop**

As stated, Algorithm 3 is very general as we have not specified the sequences $\{u_k\}$ and $\{s_k\}$ that determine the number seen and unseen data points datapoints processed at iteration k. These choices give rise to various algorithms, some known and some new. For example, the stochastic gradient method (3.2) is obtained by setting $s_k = 0, u_k = 1$, and

dynamic sampling methods are obtained by setting $s_k = 0$ and letting u_k be an increasing sequence. The purpose of allowing additional samples ($u_k > 0$) at every iteration is to achieve an increasingly accurate approximation y_k of the gradient $\nabla F(x_k)$.

Various other examples of $\{u_k\}$ and $\{s_k\}$ that we have considered are discussed in section 3.3. Two natural choices are $u_k = O(k)$ and $u_k = O(a^k)$ for some $a > 1$. We see in the next section that linear convergence in the expected value of $F(x_k)$ can be established for strongly convex functions.

The main questions we want to answer in the rest of the chapter is whether the additional flexibility of the EGR algorithm can be beneficial in practice, and whether it is supported by a convergence theory.

3.1.1. Control Variate Motivation for (3.12)

To motivate the formula (3.12), we follow the argument given in [**12**] based on the control variate method of variance reduction. Let X and Y be scalar random variables that are correlated. When the goal is to estimate $\mathbb{E}[X]$, an unbiased estimator of this quantity is

$$Z_\beta = X - \beta(Y - \mathbb{E}Y),$$

where $\beta \in \mathbb{R}$ is a control parameter parameter. The variance is given by

$$Var(Z_\beta) = Var(X) + \beta^2 Var(Y) - 2\beta Cov(X, Y).$$

When X and Y are correlated, the variance of the estimator is smaller than of $Var(X)$. Choosing X to be components of the sample gradient randomly chosen from the previously seen points, and Y to be components of a previously computed gradient at that sample

point, an unbiased estimate of the true gradient, with a (hopefully) reduced variance is

$$\nabla f^j(\phi_k^j) - \beta_k \left(\nabla f^j(\phi_{k-1}^j) - \frac{1}{t_k} \sum_{i=1}^{t_k} \nabla f^i(\phi_{k-1}^i) \right).$$

When multiple such samples are considered, along with new sample gradients from unseen points, we can define an unbiased step of the form

$$y_k = \frac{1}{s_k + u_k} \left(\sum_{j \in S_k} \left[\nabla f^j(\phi_k^j) - \beta_k \left(\nabla f^j(\phi_{k-1}^j) - \frac{1}{t_k} \sum_{i=1}^{t_k} \nabla f^i(\phi_{k-1}^i) \right) \right] + \sum_{j \in U_k} \nabla f^j(\phi_k^j) \right).$$

Step (3.12) employs this formula with $\beta_k = 1$. When $\beta_k = 0$, the variance of the samples seen so far is not reduced. In general, the parameter β_k can be tuned to the application.

3.2. Convergence Analysis

We now study the convergence properties of an instance of Algorithm 1 in which the number of new samples u_k grows geometrically and y_k is given by (3.10). At the beginning of the k-th iteration, the algorithm has t_k gradients in storage indexed by the set $T_k = \{1, \ldots, t_k\}$. We make the following assumptions about the construction of the sets S_k and U_k, and about the objective function F.

Assumptions

There needs to be some rounding to integers. In what way does that affect the analysis?

    A.1. Let $s_k = u_k = \eta(1+\eta)^{k-1}$, with $\eta \in (0,1)$, and $u_0 = 1$.

    A.2. Each sample function f^i has a Lipschitz continuous gradient with parameter L_i, and the objective function F given in (3.1) has a Lipschitz continuous gradient with parameter L_F. This enables us to define a bound $L < \infty$ on the average of

constants L_i:

$$(3.14) \qquad L = \max\left\{ L_F, \ \max_k \left\{ \frac{1}{t_k} \sum_{i=1}^{t_k} L_i \right\} \right\}.$$

A.3. F is strongly convex witha strong convexity parameter μ.

A.4. The trace of the variance of individual sample gradients is uniformly bounded. That is, there exists a parameter v such that

$$\text{tr}(Var(\nabla f^i(x))) \leq v^2, \ \forall x.$$

Preliminary Definitions

In order to prove the linear convergence of the algorithm, we must define additional quantities that evolve as the algorithm progresses: the gradient error e_k, and the variance bound σ_k.

We first consider the distance between the direction y_k and the gradient $\nabla F_{T_k}(x)$ of the sample approximation (3.6). For this purpose, we define the individual errors for all samples observed up to, and including, iteration k:

$$(3.15) \qquad e_k^i = \nabla f^i(\phi_k^i) - \nabla f^i(x_k), \quad i \in T_k \cup U_k.$$

It follows from (3.13) that $e_k^i = 0$ for $i \in S_k \cup U_k$.

Since we can rewrite (3.10) as

$$(3.16) \qquad y_k = \frac{1}{t_{k+1}} \left(\sum_{i \in S_k} \nabla f^i(x_k) + \sum_{i \in (T_k - S_k)} \nabla f^i(\phi^i_{k-1}) + \sum_{i \in U_k} \nabla f^i(x_k) \right)$$

$$(3.17) \qquad = \frac{1}{t_{k+1}} \left(\sum_{i \in S_k} \nabla f^i(x_k) + \sum_{i \in (T_k - S_k)} \nabla f^i(\phi^i_k) + \sum_{i \in U_k} \nabla f^i(x_k) \right)$$

$$(3.18) \qquad = \frac{1}{t_{k+1}} \left(\sum_{i \in U_k \cup T_k} \nabla f^i(x_k) + \sum_{i \in (T_k - S_k)} [\nabla f^i(\phi^i_k) - \nabla f^i(x_k)] \right),$$

it follows that

$$(3.19) \qquad y_k = \nabla F_{T_{k+1}}(x_k) + \frac{1}{t_{k+1}} \sum_{i=1}^{t_k} e^i_k.$$

By assumption A.1, we have that

$$(3.20) \qquad t_k = \sum_{i=0}^{k-1} u_i = (1+\eta)^{k-1}, \quad \text{and} \quad t_{k+1} = \sum_{i=0}^{k} u_i = (1+\eta)^k = (1+\eta)t_k.$$

At each iteration k, an index $i \in T_k = \{1, \ldots, t_k\}$ is included in S_k with probability $\frac{s_k}{t_k} = \frac{\eta(1+\eta)^{k-1}}{(1+\eta)^{k-1}} = \eta$. So, from (3.13) we have that

$$\phi^i_k = \begin{cases} x_k & \text{if } i \in U_k, \\ x_k & \text{with probability } \eta, \text{ if } i \in T_k, \\ \phi^i_{k-1} & \text{with probability } 1 - \eta, \text{ if } i \in T_k, \end{cases}$$

and therefore, for $i \in T_k$

$$\|e_k^i\| = \begin{cases} 0 & \text{with probability } \eta, \\ \|e_{k-1}^i + \nabla f^i(x_{k-1}) - \nabla f^i(x_k)\| & \text{with probability } 1 - \eta. \end{cases}$$

Then, using Assumption A.2.,

(3.21)

$$\mathbf{E}_k[\|e_k^i\|] = (1-\eta)\|e_{k-1}^i + \nabla f^i(x_{k-1}) - \nabla f^i(x_k)\| \le (1-\eta)\|e_{k-1}^i\| + (1-\eta)L_i\|x_{k-1} - x_k\|,$$

where $\mathbf{E}_k$ is the expectation over all choices of S_k at iteration k. We define a bound on all individual errors e_k^i to be

(3.22)
$$e_k = \frac{1}{t_{k+1}} \sum_{i=1}^{t_{k+1}} \|e_k^i\| = \frac{1}{t_{k+1}} \sum_{i=1}^{t_k} \|e_k^i\|,$$

where the last equality follows from $t_{k+1} = t_k + u_k$ and the fact that $e_k^i = 0$ for $i \in U_k$. Therefore, we have from (3.19) that

(3.23)
$$\|y_k\| \le \|\nabla F_{T_{k+1}}(x_k)\| + e_k.$$

Next, we estimate the difference between the gradients of the sample approximation $F_{T_k}(x)$ and the objective function F. Let $\mathbf{E}[\cdot]$ denote the expectation with respect to all

the choices of the sets U_ℓ, S_ℓ for $\ell = 1, 2 \ldots$. By Jensen's inequality,

$$
\begin{aligned}
(\mathbf{E}[\|\nabla F_{T_k}(x) - \nabla F(x)\|])^2 &\le \mathbf{E}[\|\nabla F_{T_k}(x) - \nabla F(x)\|^2] \\
&= \mathbf{E}[\mathrm{tr}(\nabla F_{T_k}(x) - \nabla F(x))(\nabla F_{T_k}(x) - \nabla F(x))^T] \\
&= \mathrm{tr}(\mathrm{Var}(\nabla F_{T_k}(x))) \\
&= \mathrm{tr}\left(\mathrm{Var}\left(\frac{1}{t_k}\sum_{i=1}^{t_k}\nabla f^i(x)\right)\right) \\
&\le \frac{1}{t_k}\mathrm{tr}(\mathrm{Var}(\nabla f^i(x))) \le \frac{v^2}{t_k},
\end{aligned}
$$

where the last equality follows from Assumption A.4, and the fact that ξ_i are independent. If we define

$$(3.24) \qquad\qquad \sigma_{k-1} = \sqrt{v^2/t_k}$$

we can conclude that

$$(3.25) \qquad\qquad \mathbf{E}\|\nabla F_{T_k}(x) - \nabla F(x)\| \le \sigma_{k-1} \qquad \text{for all } x \in \mathbb{R}^n.$$

Now, since the number of observed data grows geometrically, i.e. $t_{k+1} = (1+\eta)t_k$, the bound σ_k decays geometrically,

$$(3.26) \qquad\qquad \sigma_k = \sqrt{\frac{v^2}{(1+\eta)t_k}} = \sqrt{\frac{1}{1+\eta}}\sigma_{k-1}.$$

Lemma 3.2.1 describes the joint evolution of three quantities: e_k, σ_k and $\|x_k - x_*\|$. The sum of these 3 quantities can be seen as a Lyapunov function. We recall that $\mathbf{E}[\cdot]$ denotes the expectation with respect to all the choices of the sets U_ℓ, S_ℓ, $\ell = 1, 2, \ldots$

Lemma 3.2.1. *Let $\{x_k\}$ be the iterates produced by Algorithm 3, where y_k is given by (3.10), and suppose that Assumptions A1 - A4 hold. Then, provided that $\alpha \le \frac{1}{L}$,*

$$
(3.27) \qquad \begin{pmatrix} \frac{1}{L}\mathbf{E}[\mathbf{E}_k[e_k]] \\ \mathbf{E}[\|x_k - x_*\|] \\ \sigma_k \end{pmatrix} \le M \begin{pmatrix} \frac{1}{L}\mathbf{E}[e_{k-1}] \\ \mathbf{E}[\|x_{k-1} - x_*\|] \\ \sigma_{k-1} \end{pmatrix},
$$

where

$$
(3.28) \qquad M = \begin{pmatrix} (\frac{1-\eta}{1+\eta})(1+\alpha L) & (\frac{1-\eta}{1+\eta})\alpha L & (\frac{1-\eta}{1+\eta})\alpha \\ \alpha L & 1 - \alpha\mu & \alpha \\ 0 & 0 & \sqrt{\frac{1}{1+\eta}} \end{pmatrix},
$$

and x_* is the unique minimizer of (3.1).

In general, the presentation of this proof should be significantly improved. It took me a very long time to follow it. More intermediate steps and grouping terms more consistently would help.

Proof. By using (3.22), (3.21), (3.8), (3.14), (3.19), (3.23), (3.25),

$$\mathbf{E}[\mathbf{E}_k[e_k]] = \frac{1}{t_{k+1}} \sum_{i=1}^{t_k} \mathbf{E}[\mathbf{E}_k[\|e_k^i\|]]$$

$$\leq \frac{1}{t_{k+1}} \sum_{i=1}^{t_k} \left((1-\eta)\mathbf{E}\|e_{k-1}^i\| + (1-\eta)L_i\mathbf{E}\|x_{k-1} - x_k\|\right)$$

$$\leq (1-\eta)\frac{1}{t_{k+1}} \left(\sum_{i=1}^{t_k} \mathbf{E}\|e_{k-1}^i\|\right) + (1-\eta)\frac{1}{t_{k+1}}\mathbf{E}\|\alpha y_{k-1}\| \sum_{i=1}^{t_k} L_i$$

$$\leq (\frac{1-\eta}{1+\eta})\mathbf{E}\left(\frac{1}{t_k} \sum_{i=1}^{t_k} \|e_{k-1}^i\|\right) + (\frac{1-\eta}{1+\eta})L\alpha\left(\mathbf{E}\|\nabla F_{T_k}(x_{k-1})\| + \mathbf{E}[e_{k-1}]\right)$$

$$\leq (\frac{1-\eta}{1+\eta})(1+L\alpha)\mathbf{E}[e_{k-1}] + (\frac{1-\eta}{1+\eta})L\alpha\mathbf{E}\|\nabla F_{T_k}(x_{k-1})\|$$

$$= (\frac{1-\eta}{1+\eta})(1+L\alpha)\mathbf{E}[e_{k-1}] + (\frac{1-\eta}{1+\eta})L\alpha\mathbf{E}\|\nabla F_{T_k}(x_{k-1}) - \nabla F(x_{k-1}) + \nabla F(x_{k-1}) - \nabla F(x_*)\|$$

$$\leq (\frac{1-\eta}{1+\eta})(1+L\alpha)\mathbf{E}[e_{k-1}] + (\frac{1-\eta}{1+\eta})L\alpha\sigma_{k-1} + (\frac{1-\eta}{1+\eta})L\alpha\mathbf{E}\|\nabla F(x_{k-1}) - \nabla F(x_*)\|$$

Therefore,

$$(3.29) \quad \mathbf{E}[\mathbf{E}_k[e_k]] \leq (\frac{1-\eta}{1+\eta})(1+L\alpha)\mathbf{E}[e_{k-1}] + (\frac{1-\eta}{1+\eta})L\alpha\sigma_{k-1} + (\frac{1-\eta}{1+\eta})L^2\alpha\mathbf{E}\|x_{k-1} - x_*\|$$

Next, let us consider $\mathbf{E}[\|x_k - x_*\|]$:

$$\mathbf{E}\|x_k - x_*\| = \mathbf{E}\|x_{k-1} - x_* + x_k - x_{k-1}\|$$

$$= \mathbf{E}\|x_{k-1} - x_* - \alpha y_{k-1}\|$$

$$= \mathbf{E}\left\|x_{k-1} - x_* - \alpha\left(\nabla F_{T_k}(x_{k-1}) + \frac{1}{t_k}\sum_{i=1}^{t_k} e_{k-1}^i + \nabla F(x_{k-1}) - \nabla F(x_{k-1})\right)\right\|$$

$$\leq \mathbf{E}\|x_{k-1} - x_* - \alpha\nabla F(x_{k-1})\| + \alpha\mathbf{E}\|\nabla F_{T_k}(x_{k-1}) - \nabla F(x_{k-1})\| + \alpha\mathbf{E}[e_{k-1}]$$

$$= \mathbf{E}\left\|\left(I - \alpha\int_{t=0}^{1}\nabla^2 F(tx_{k-1} + (1-t)x_*)\right)(x_{k-1} - x_*)\right\| + \alpha\sigma_{k-1} + \alpha\mathbf{E}[e_{k-1}],$$

Where we use the fact that $\alpha \leq \frac{1}{L}$. By Assumption A.4 we get

$$(3.30) \qquad \mathbf{E}\|x_k - x_*\| \leq (1 - \alpha\mu)\mathbf{E}\|x_{k-1} - x_*\| + \alpha\sigma_{k-1} + \alpha\mathbf{E}[e_{k-1}].$$

Now, gathering (3.29), (3.30), and (3.26) we obtain (3.27). □

The next Lemma describes a bound on the spectral radius of matrix M. This bound provides the rate of convergence of EGR as will be shown in Theorem 3.2.3, and is further elaborated in Remark 3.2.5.

Lemma 3.2.2. *Choose the constant steplength*

$$(3.31) \qquad \alpha = \min\left\{\beta\left(\frac{2\eta}{1-\eta}\right)\frac{\mu}{L}\frac{1}{\mu + L}, \frac{1}{L}\right\}, \quad \text{for some } \beta \in (0,1).$$

The spectral radius $\lambda_{\max}^M$ of matrix M in (3.28) then satisfies

$$(3.32)$$
$$\sqrt{\frac{1}{\eta+1}} < \lambda_{\max}^M \leq \max\left\{\sqrt{\frac{1}{\eta+1}}, 1 - \frac{\alpha\left(2\eta\mu + \alpha(\eta-1)L^2 + \alpha(\eta-1)\mu L\right)}{\alpha((\eta+1)\mu - (1-\eta)L) + 2\eta}\right\} < 1.$$

Proof. The 3×3 matrix (3.28) has three eigenvalues, we have $\lambda_3^M = \sqrt{\frac{1}{1+\eta}}$, and λ_1^M, λ_2^M are the eigenvalues of upper-left 2×2 block

(3.33)
$$B = \begin{pmatrix} (\frac{1-\eta}{1+\eta})(1 + \alpha L) & (\frac{1-\eta}{1+\eta})\alpha L \\ \alpha L & 1 - \alpha \mu \end{pmatrix}.$$

Let
$$A = \begin{pmatrix} (\frac{1-\eta}{1+\eta})(1 + \alpha L) - 1 & (\frac{1-\eta}{1+\eta})\alpha L \\ \alpha L & -\alpha \mu \end{pmatrix}$$

so that
$$B = I + A.$$

We will first show that both eigenvalues of A, λ_1^A, λ_2^A, are negative.

Case I. In (3.31), $\alpha = \beta \left(\frac{2\eta}{1-\eta}\right) \frac{\mu}{L} \frac{1}{\mu+L}$.

$$A_{11} = (\frac{1-\eta}{1+\eta})(1 + \alpha L) - 1 = -\frac{2\eta(-\beta\mu + \mu + L)}{(\eta+1)(\mu+L)} < 0.$$

So, both $A_{11} < 0$ and $A_{22} < 0$, yielding $\text{tr}(A) < 0$. On the other hand,

$$\det(A) = ((\frac{1-\eta}{1+\eta})(1 + \alpha L) - 1)(-\alpha\mu) - (\frac{1-\eta}{1+\eta})\alpha^2 L^2$$

$$= \frac{\alpha\,(2\eta\mu - \alpha(1-\eta)L^2 - \alpha(1-\eta)\mu L)}{\eta + 1}$$

$$= \frac{4(1-\beta)\beta\eta^2\mu^2}{(1-\eta^2)\,L(\mu+L)}$$

$$> 0$$

Case II. In (3.31), $\alpha = \frac{1}{L}$. Note that in this case

$$(3.34) \qquad \beta\left(\frac{2\eta}{1-\eta}\right)\frac{\mu}{\mu+L} \geq 1 \quad \Leftrightarrow \quad (\frac{1-\eta}{1+\eta}) \leq \left(\frac{1+\kappa}{\beta}+1\right)^{-1} < (\kappa+2)^{-1},$$

where $\kappa = \dfrac{L}{\mu}$. Then,

$$\mathrm{tr}(A) < A_{11} = (\frac{1-\eta}{1+\eta})(2) - 1 \leq 2\,(\kappa+2)^{-1} - 1 < 0,$$

and

$$\begin{aligned}
\det(A) &= ((\frac{1-\eta}{1+\eta})(2) - 1)(-\frac{1}{L}\mu) - (\frac{1-\eta}{1+\eta}) \\
&= (1 - (\frac{1-\eta}{1+\eta})(2))(\kappa) - (\frac{1-\eta}{1+\eta}) \\
&= \kappa + (-2\kappa - 1)(\frac{1-\eta}{1+\eta}) \\
&\geq \kappa + \frac{-2\kappa - 1}{\kappa + 2} \\
&\geq 0
\end{aligned}$$

Therefore we conclude that $\lambda_1^A, \lambda_2^A < 0$. Number them so that $\lambda_1^A \geq \lambda_2^A$. To bound λ_1^A above note that

$$\frac{\det(A)}{\mathrm{tr}(A)} = \frac{\lambda_1^A \lambda_2^A}{\lambda_1^A + \lambda_2^A} > \lambda_1^A$$

since $\lambda_2^A/(\lambda_1^A + \lambda_2^A) \leq 1$. Therefore,

(3.35)
$$\lambda_{\max}^A < \frac{\det(A)}{\mathrm{tr}(A)} = -\frac{-\alpha(\frac{1-\eta}{1+\eta})(\mu(1+\alpha L) + \alpha L^2) + \alpha\mu}{1 - (\frac{1-\eta}{1+\eta})(1+\alpha L) + \alpha\mu} < -\alpha\mu\frac{1 - (\frac{1-\eta}{1+\eta})(1+\alpha L(1+\kappa))}{1 + \alpha\mu}.$$

$$\lambda_{\max}^A < \frac{\det(A)}{\operatorname{tr}(A)} = -\frac{\alpha\left(2\eta\mu + \alpha(\eta-1)L^2 + \alpha(\eta-1)\mu L\right)}{(\eta+1)\left(\alpha\mu - \frac{(1-\eta)(\alpha L+1)}{\eta+1} + 1\right)}$$

$$= -\frac{\alpha\left(2\eta\mu + \alpha(\eta-1)L^2 + \alpha(\eta-1)\mu L\right)}{\alpha((\eta+1)\mu - (1-\eta)L) + 2\eta}$$

$$< 0$$

Since $\lambda_{\max}^M = \max\{\sqrt{\frac{1}{1+\eta}}, \lambda_1^M, \lambda_2^M\}$ and we have $\lambda_1^M = 1 + \lambda_1^A$, $\lambda_2^M = 1 + \lambda_2^A$, the upper and lower bound result follows.

Theorem 3.2.3. *Suppose Assumptions A.1-A.4 hold, and y_k is computed with (3.10). Then, $\{\mathbf{E}\|x_k - x_*\|\}$ produced by Algorithm 3 converges to zero R-linearly.*

Proof. By Lemma 1,

$$(3.36) \qquad \begin{pmatrix} \frac{1}{L}\mathbf{E}[e_k] \\ \mathbf{E}[\|x_k - x_*\|] \\ \sigma_k \end{pmatrix} \leq M^k \begin{pmatrix} \frac{1}{L}e_0 \\ \|x_0 - x_*\| \\ \sigma_0 \end{pmatrix}$$

To study the norm of M we note that since the upper left 2×2 matrix is positive, it has 2 distinct eigenvalues, and thus 2 linearly independent eigenvectors. If we append a zero to these vectors we have 2 independent eigenvectors of M which are clearly independent of the eigenvector corresponding to ρ. If we define S to be the matrix whose columns are these 3 eigenvectors, we have that $S^{-1}MS = D$ where D is diagonal. Now if we define the weighted vector norm

$$\|x\|_S \equiv \|S^{-1}x\|_2,$$

then the corresponding induced matrix norm of M is

$$\|M\|_S \equiv \|S^{-1}MS\|_2 = \|D\|_2 = \lambda_{max}^M.$$

It follows immediately that

(3.37)
$$\left\| \begin{pmatrix} \frac{1}{L}\mathbf{E}[e_k] \\ \mathbf{E}[\|x_k - x_*\|] \\ \sigma_k \end{pmatrix} \right\|_S \leq (\lambda_{max}^M)^k \left\| \begin{pmatrix} \frac{1}{L}e_0 \\ \|x_0 - x_*\| \\ \sigma_0 \end{pmatrix} \right\|_S$$

where λ_{max}^M satisfies (3.32). □

It is worth noting that the above analysis provides a simple proof of the R-linear convergence of the SAG algorithm, since SAG is equivalent to Algorithm 3 applied to a finite population, with $t_k = m$, $u_k = 0$ and s_k is a constant k.

Corollary 3.2.4. *Suppose$\{x_k\}$ is produced by the SAG method applied to a population of size m, i.e. Algorithm 3 where $s_k = 1$ and $u_k = 0$ for all k, with y_k is computed with (3.10). Suppose Assumptions A.2-A.4 hold. Then, $\{\mathbf{E}\|x_k - x_*\|\}$ converges to zero R-linearly.*

"We also have eta=0"

Proof. Since for this problem $F(x) = \frac{1}{m}\sum_{i=1}^{m} f^i(x)$, we have $\sigma_k = 0$ for all k. It follows from Lemma 3.2.1 that

(3.38)
$$\begin{pmatrix} \frac{1}{L}\mathbf{E}[\mathbf{E}_k[e_k]] \\ \mathbf{E}[\|x_k - x_*\|] \end{pmatrix} \leq M \begin{pmatrix} \frac{1}{L}\mathbf{E}[e_{k-1}] \\ \mathbf{E}[\|x_{k-1} - x_*\|] \end{pmatrix}$$

where

(3.39)
$$M = \begin{pmatrix} (1-\eta)(1+\alpha L) & (1-\eta)\alpha L \\ \alpha L & 1 - \alpha\mu. \end{pmatrix},$$

Then, by Lemma 3.2.2, if we use the steplength

(3.40)
$$\alpha = \min\left\{ \beta\left(\frac{2\eta}{1-\eta}\right)\frac{\mu}{L}\frac{1}{\mu+L}, \frac{1}{L}\right\}, \quad \text{for some } \beta \in (0,1),$$

then the spectral radius $\lambda_{\max}^M$ of the matrix M satisfies

(3.41)
$$\lambda_{\max}^M \leq 1 - \frac{\alpha\left(2\eta\mu + \alpha(\eta-1)L^2 + \alpha(\eta-1)\mu L\right)}{\alpha((\eta+1)\mu - (1-\eta)L) + 2\eta}$$

It follows from (3.38) that $\{\mathbf{E}\|x_k - x_*\|\}$ converges to zero R-linearly.

$\square$

In the last remark, we specify the convergence rate of our algorithm. The result is similar to SAG for a large enough growth rate, which also depends on the condition number like $\Theta(\kappa^{-1})$, see [45].

Remark 3.2.5. *If the parameter η satisfies $(\frac{1-\eta}{1+\eta}) \leq \left((1+\kappa)\frac{\theta}{\beta} + 1\right)^{-1}$, then we have*

(3.42)
$$\lambda_{\max}^A < -\frac{(1-\beta)\theta}{(1+\beta)(\kappa+\theta)} = -\Theta(\kappa^{-1}).$$

Otherwise,

(3.43)
$$\lambda_{\max}^A < -\frac{\beta(1-\beta)(1-(\frac{1-\eta}{1+\eta}))^2}{(\frac{1-\eta}{1+\eta})\kappa(\kappa+1) + \beta} = -\Theta(\kappa^{-2}).$$

Proof. In (3.35), we place the corresponding value of α from (3.31):

Case I. $\alpha = \dfrac{\theta}{L}$.

$$-\frac{\alpha\mu}{1+\alpha\mu}\left(1-(\frac{1-\eta}{1+\eta})[1+\alpha L(1+\kappa)]\right) = -\frac{\theta}{\kappa+\theta}\left(1-(\frac{1-\eta}{1+\eta})[1+\theta(1+\kappa)]\right)$$

In light of (3.34),

$$1-(\frac{1-\eta}{1+\eta})[1+\theta(1+\kappa)] \geq 1 - \frac{\beta(1+\theta(1+\kappa))}{\beta+\theta(1+\kappa)} = \frac{(1-\beta)\theta(1+\kappa)}{\beta+\theta(1+\kappa)} > (1-\beta)\frac{1}{\beta+1}.$$

Case II. $\alpha = \beta\left(\dfrac{2\eta}{1-\eta}\right)\dfrac{\mu}{L}\dfrac{1}{\mu+L}$.

$$-\frac{\alpha\mu}{1+\alpha\mu}\left(1-(\frac{1-\eta}{1+\eta})[1+\alpha L(1+\kappa)]\right) = -\frac{\beta(1-(\frac{1-\eta}{1+\eta}))}{\beta(1-(\frac{1-\eta}{1+\eta}))+(\frac{1-\eta}{1+\eta})\kappa(\kappa+1)}(1-(\frac{1-\eta}{1+\eta}))(1-\beta)$$

$$< -\frac{\beta(1-\beta)(1-(\frac{1-\eta}{1+\eta}))^2}{\beta+(\frac{1-\eta}{1+\eta})\kappa(\kappa+1)}.$$

$\square$

3.3. Implementation

In this section we describe choices for the sequences $\{s_k\}$ and $\{u_k\}$, and the characteristics of the resulting methods. We also present an efficient way of implementing Algorithm 3, and make a few comments about techniques for reducing storage. By carefully selecting sequences u_k and s_k in Algorithm 1, we arrive at some known algorithms, these are mentioned in the only-add and only-update sections below. Some additional possible choices are outlined in Table 3.1.

There are some implicit restrictions in the input choices of Algorithm 3. Clearly, we must have that $s_0 = 0$ and $u_0 > 0$, and thus $t_1 = u_0$. By construction, $t_k = \sum_{i=0}^{k-1} u_i$ for

$k > 0$, and we must have that $s_t \leq t_k$. For finite datasets of size m, u_k must become 0 once the whole training set has been exhausted. Table 3.1, lists five methods that can generated by the EGR framework; two of these are well known. A short discussion of these algorithms is given below.

Algorithm	u_k	t_k	$\frac{u_k}{t_k}$	s_k for $k > 0$	$\frac{s_k}{t_k}$
Only-add	$*$	$*$	$*$	0	0
Only-update	$\begin{array}{ll} m & \text{if } k = 0 \\ 0 & \text{if } k > 0 \end{array}$	m	0	$*$	$*$
EGR-lin	r	rk	$\frac{1}{k}$	r	$\frac{1}{k}$
EGR-quad	$r(k+1)$	$\frac{rk(k+1)}{2}$	$\frac{2}{k}$	rk	$\frac{2}{k+1}$
EGR-exp	$\begin{array}{ll} 1 & \text{if } k = 0 \\ \frac{1}{r-1}\left(1 + \frac{1}{r-1}\right)^{k-1} & \text{if } k > 0 \end{array}$	$\left(1 + \frac{1}{r-1}\right)^{k-1}$	$\frac{1}{r-1}$	$\frac{1}{r-1}\left(1 + \frac{1}{r-1}\right)^{k-1}$	$\frac{1}{r-1}$

Table 3.1. Methods obtained from Algorithm 3 by specifying the sequences $\{s_k\}$ and $\{u_k\}$. A $*$ means that any (integer) value can be chosen. It is understood that the `ceiling` function $\lceil \cdot \rceil$ should be applied to some entries to ensure the resulting values of u_k, s_k are always integers.

It might be worth pointing out that larger r in exp means smaller batches, in contrast to the other options. Or replace the current 1/(r-1) with r

`Only-add` algorithms do not store old gradients. They include the stochastic gradient method ($u_k = 1$) and dynamic sampling methods, where u_k typically grows as a function of k. Both methods are very successful in practice and the convergence of sequence $\{F(x_k)\}$ to F^* is well understood.

`Only-update` A few well known methods fall into this category, including the classic gradient descent algorithm, SAG, and SAGA. These algorithms work on a static batch

of size m. Most of the proven results are about the quantity F_m^*, but some also estimate $\|x_m^K - x^*\|$.

`egr-lin`. In this version of the EGR method employs a constant number of replacements and a constant number of new samples at each iteration, which give rise to a linear growth in t_k.

`egr-quad`. Here the number of replacements and new samples both grow linearly at each iteration. This results in a quadratic growth in t_k.

`egr-exp`. This choice ensures that the sequence $\{t_k\}$ grows by a fixed factor at every iteration. This strategy starts with a large initial batch of size $c(r-1)$ for u_0, and then computes a constant fraction of t_k and resamples that amount, thus growing at an exponential rate.

3.3.1. Implemented Version of the Algorithm

The way Algorithm 3 is stated suggest computing a sum over all $\{1, \ldots, t_k\}$ at every iteration. This costly computation can be avoided by storing the current sum, and updating it accordingly at every iteration. Our implementation in Julia is available at `https://github.com/stefanks/EGR.jl`.

Algorithm 4 EGR Algorithm (Implementation Version)

$\qquad$ **Input:** $\{s_k\}$, $\{u_k\}$, α, x

1: $T = 0$, $k = 0$, $A = 0$

2: **loop**

3: $\qquad U_k = \{I + 1, \ldots, I + u_k\}$

4: $\qquad$ For all $j \in U_k$ draw ξ^j from the distribution P

5: $\qquad S_k = $ sample of s_k indices from $\{1, \ldots, I\}$

6: $\qquad B = \sum_{j \in S} g^j$

7: $\qquad g^j = \nabla f^j(x)$ for all $j \in S \cup U$

8:

(3.44)
$$y = \frac{\frac{s_k}{T} A - B + \sum_{j \in S \cup U} g^j}{s_k + u_k} \text{ EGR-SAGA} \quad \text{or} \quad y = \frac{A - B + \sum_{j \in S \cup U} g^j}{I + u_k} \text{ EGR-SAG}$$

9: $\qquad x = x - \alpha y$

10: $\qquad T = T + u_k$

11: $\qquad A = A - B + \sum_{j \in S \cup U} g^j$

12: $\qquad k = k + 1$

13: **end loop**

We conclude this section with a few brief comments on techniques that can be employed to reduce storage in the aggregated gradient methods, and by extension in the EGR method.

3.3.2. Memory Reduction Techniques

Aggregated gradient methods such as SAG and SAGA require storing m gradients to compute each iteration. In machine learning applications, the sample gradients can be stored compactly, see the discussion in the next section. We propose an alternative way of reducing this memory requirement. The strategy, which we call *chunking*, can be extended to the EGR algorithm. We expect performance to deteriorate compared to full memory implementation, with the advantage of have a slimmer memory requirement.

3.3.2.1. Compact Storage. In machine learning applications, where the sample functions f^i are loss function, they often have the property that

$$(3.45) \qquad\qquad f^i(x) = g(W_i x),$$

for some function g and $c \times n$ matrices W_i. In this case, the gradient of f with respect to x is computed as

$$(3.46) \qquad\qquad \nabla f^i(x) = W_i^T g'(W_i x),$$

where $g'(W_i x)$ is a vector of length c. Therefore, storing the vector $g'(W_i x)$ is sufficient to reconstruct the original gradient at x. For binary classification problems with certain loss functions, such as the cross entropy loss, the matrix W_i is an $1 \times n$ vector, reducing the storage requirement to a scalar. Note that the addition of an $L2$ regularizer destroys the property (3.45), thereby making compact storage impossible.

3.3.2.2. Chunking. Assume that s_k and u_k are divisible by some C for every k. In this case, we can introduce the concept of chunking, or mini-batch replacement and storing.

This requires grouping the sample functions into chunks of size C, indexed by j. Instead of storing the values of each $\nabla f^i(\phi_i^k)$, we store appropriate sample gradient averages $\frac{1}{C} \sum_{i \in Chunk_j} \nabla f^i(x)$. Since an average of sample gradients is not of the form (3.45), the compact storage property cannot be used. On the other hand, for large enough chunks this might not a drawback at all, since the sheer reduction in the number of vectors to be stored may make the storage requirement less demanding even compared to using compact forms. Clearly, chunking is not appropriate when storing averages of gradients is prohibitively expensive.

Note that chunking and *batching* are different algorithms. Specifically, a batched version of EGR would simply be a version with s_k or u_k being greater than 1, without reducing memory requirements.

The following algorithms are implementations of the chunking idea in case of the original SAG and SAGA algorithms, as well as to the proposed EGR algorithm.

Algorithm 5 SAG - Chunking

Input: Initial iterate x^0, chunk size C, steplength α, number of datapoints m, g_i^{-1} for

$$\text{each } i \in \{1, \ldots, \tfrac{m}{C}\}$$

1: **loop** $k = \{0, 1, \ldots\}$

2:    Sample i from $\{1, \ldots, \tfrac{m}{C}\}$

3:    chunk $i = \{(i-1)C + 1, \ldots, iC\}$

4:    $g_i^k = \frac{1}{C} \sum_{j \in \text{chunk } i} \nabla f^j(x_k)$

5:    $g_j^k = g_j^{k-1}$ for $j \neq i$

6:    $y_k = \frac{C}{m} \sum_{j=1}^{\frac{m}{C}} g_j^k$

7:    $\theta^{k+1} = x_k - \alpha y_k$

8: **end loop**

Algorithm 6 SAGA - Chunking

Input: Initial iterate x^0, chunk size C, steplength α, number of datapoints m, g_i^{-1} for

$$\text{each } i \in \{1, \ldots, \tfrac{m}{C}\}$$

1: **loop** $k = \{0, 1, \ldots\}$

2:    Sample i from $\{1, \ldots, \tfrac{m}{C}\}$

3:    chunk $i = \{(i-1)C + 1, \ldots, iC\}$

4:    $g_i^k = \frac{1}{C} \sum_{j \in \text{chunk } i} \nabla f^j(x_k)$

5:    $g_j^k = g_j^{k-1}$ for $j \neq i$

6:    $y_k = g_i^k - g_i^{k-1} + \frac{C}{m} \sum_{j=1}^{\frac{m}{C}} g_j^{k-1}$

7:    $\theta^{k+1} = x_k - \alpha y_k$

8: **end loop**

This does not seem to be complete
and needs some serious editing

Algorithm 7 EGR - Chunking

Input: Sequences $\{s_k\}$, $\{u_k\}$, steplength α, chunk size C, and initial iterate x_0

1: Set $t_0 = 0$

2: **loop** $k = 0, 1, 2, , \ldots$

3:     $U_k = \{t_k + 1, \ldots, t_k + u_k\}$

4:     For each $j \in U_k$ draw C samples $\xi_j^1, \ldots, \xi_j^C$ from the distribution P, and define

$Chunk_j = \{\}$

5:     $S_k =$ sample of s_k indices from $\{1, \ldots, t_k\}$

6:     Evaluate $\nabla f^j(x_k) = \nabla f(x_k, \xi_j)$ for $j \in S_k \cup U_k$

7:     Compute y_k by (3.10) or (3.12)

8:

$$(3.47) \quad \text{Set } \phi_k^i = \begin{cases} x_k & \text{for all } i \in S_k \cup U_k \\ \\ \phi_{k-1}^i & \text{otherwise} \end{cases}$$

9:     $x_{k+1} = x_k - \alpha y_k$

10:     $t_{k+1} = t_k + u_k$

11: **end loop**

3.4. Numerical Tests

The purpose of this section is to qualitatively measure the performance of EGR in the early iterations of the algorithm, when the number of evaluated sample gradients is smaller than the number of available training points m, i.e. the single-epoch mode. This experimental mode allows us to make conclusions about the performance of our algorithm

when used on a problem with a very large training set, and the "oracle" or "streaming" case, where new samples from P are available on demand. Both cases are common in practice. Therefore, a direct comparison of EGR with aggregated gradient methods such as SAG and SAGA is not appropriate, since any iteration of these methods requires pre-computed sample gradients ∇f^i for each function f^i where $i \in \{1, \ldots, m\}$. Instead, we test different growth sequences of EGR, including the Stochastic Gradient (SG) and the Dynamic Sample Size (DSS) methods, both special cases of EGR.

The EGR algorithm closely resembles the so-called *initialization heuristic* of SAG. This heuristic is mentioned alongside it's full counterpart, as a way to gradually grow the stored number of datapoints in SAG. No theoretical support is given, and the paper claims that simply running SG instead gives nearly identical performance. A possible explanation is that the function value is not evaluated at the end of the initialization phase, but instead, the performance of the later iterations is compared. We do not claim to dismiss this statement: rather, we draw new conclusions on the efficacy of using this heuristic as a stand-alone algorithm.

The `SAG-init` method, given as Algorithm 8, works as follows: at each iteration a randomly sampled datapoint is chosen from the full dataset, the corresponding sample gradient is evaluated, and this gradient replaces the old one only if it has been sampled previously. Otherwise the new gradient is stored, and the memory grows, see [45] for details. The only difference between this method and EGR is that in our framework we pre-determine the sampling from the seen and unseen sets by the sequences u_k and s_k. In `SAG-init`, the choice of sampling from S_k or U_k is random. `SAGA-init` is similar, except that it uses the SAGA direction computation.

Algorithm 8 SAG-init

Input: Steplength α, and initial iterate x_0

1: Set $T_{-1} = \{\}$, $t_{-1} = 0$

2: **loop** $k = 0, 1, 2, , \ldots$

3:      Draw j from $\{1, \ldots, m\}$

4:      Evaluate $\nabla f^j(x_k) = \nabla f(x_k, \xi_j)$

5:      **if** $j \notin T_{k-1}$ **then**

6:          $T_k = T_{k-1} + j$

7:          $t_k = t_{k-1} + 1$

8:      **else**

9:          $T_k = T_{k-1}$

10:          $t_k = t_{k-1}$

11:      **end if**

12:      Compute $y_k = \frac{1}{t_k}\left[\nabla f^j(x_k) - \nabla f^j(\phi_{k-1}^j) + \sum_{i \in T_k} \nabla f^i(\phi_{k-1}^i)\right]$

13:

$$(3.48) \quad \text{Set } \phi_k^i = \begin{cases} x_k & \text{for } i = j \\ \\ \phi_{k-1}^i & \text{otherwise} \end{cases}$$

14:      $x_{k+1} = x_k - \alpha y_k$

15: **end loop**

Algorithm 9 SAGA-init

<div align="center">Input: Steplength α, and initial iterate x_0</div>

1: Set $T_{-1} = \{\}$, $t_{-1} = 0$

2: **loop** $k = 0, 1, 2,,\ldots$

3:     Draw j from $\{1, \ldots, m\}$

4:     Evaluate $\nabla f^j(x_k) = \nabla f(x_k, \xi_j)$

5:     **if** $j \notin T_{k-1}$ **then**

6:         $T_k = T_{k-1} + j$

7:         $t_k = t_{k-1} + 1$

8:     **else**

9:         $T_k = T_{k-1}$

10:         $t_k = t_{k-1}$

11:     **end if**

12:     Compute y_k $==$ $\nabla f^j(x_k) - \nabla f^j(\phi_{k-1}^j) + \frac{1}{t_k} \sum_{i \in T_k} \nabla f^i(\phi_{k-1}^i)$

13:

(3.49) Set $\phi_k^i = \begin{cases} x_k & \text{for } i = j \\ \\ \phi_{k-1}^i & \text{otherwise} \end{cases}$

14:     $x_{k+1} = x_k - \alpha y_k$

15: **end loop**

The Streaming SVRG method would fit ideally in our experimental setup, since the motivation is identical to EGR. Unfortunately, we were not able to select parameters that

would make this algorithm competitive. Scaling algorithms such as SQN [**28**] and ADA-GRAD [**18**], while having superior, performance are outside the scope of this project.

The testing setup is as follows: all problems are publicly available binary classification datasets. Each dataset is split into two parts: a *training set* and a *testing set*. The training set constitues the information available to the algorithms through the iterations, and is used as a proxy for sampling from the distribution P. The testing set is used to assess the effectiveness of the algorithms compared, which we do using the *test function value*, which is defined as (3.4) using the testing set. Training points constitute 75% of the total number of datapoints. We use a cross-entropy loss function with L2 regularization. The L2 regularization parameter is always $\frac{1}{\text{num training points}}$.

The basis for comparison of the different methods is the number of gradient evaluations: this is a fair estimate of the total work expended for the methods tested.

3.4.1. Test Problems

The datasets are all dense, with approximately evenly split -1 and +1 labels. The dataset dimensions are summarized in Table 3.2.

Dataset	Num Features	Num Datapoints
alphaGood	500	500,000
MNIST	768	60,000
myrand	50	700,000
SUSY	18	5,000,000

Table 3.2. Dataset statistics

- **alphaGood** is a scaled version of the alpha dataset from `http://leon.bottou.org/projects/sgd`. The features are scaled to have zero mean and unit variance

- **MNIST** is a popular digit recognition dataset. <mark>We use the digit 4 as the positive label and the other digits as the negative label.</mark>

- **myrand** is an artificial dataset generated from two different multinomial Gaussian distributions.

- **SUSY** is a classification dataset from `https://archive.ics.uci.edu/ml/datasets/SUSY`, with the positive label signifies particle detection.

3.4.2. Results

We conducted three sets of experiments, each with a different objective.

- **Comparison of EGR Growth Rates** ~~EGR growth rates comparison.~~ We test the empirical performance of the three suggested growth rates. Suggestions as to the best choice of parameters for any given problem are made. The two EGR versions, EGR-SAG and EGR-SAGA are also compared.

- **Comparison with SG and Dynamic Sampling** We test the relative performance of our methods with the SG method. Stepsize tuning to different goals is discussed in detail. We also demonstrate the difference in performance when identical growth rates are used in EGR vs DSS methods. A discussion of the benefits of using/updating history follows.

- **Comparison with SAG-init Methods** A final comparison of a reasonably chosen EGR implementation with some competing algorithms such as SAG-init, SAGA-init, and SG.

3.4.2.1. Comparison of EGR Growth Rates. The three different growth rates (linear, quadratic, exponential) combined with two possible computation formulas for y_k give

rise to six methods, where each growth has an unspecified parameter. We tune it experimentally for each problem, and make observations of the effect of varying r in each case. Finally, we make suggestions on good practical choices or r.

In all plots in this section, the vertical axis is the test function value, which serves as an approximation to F. We demonstrate the continuous progression of the algorithms toward the minimizer of F. The horizontal axis is the total number of sample gradients evaluated so far. This is an approximate measure of the total work the algorithm has performed, and is more informative than the more commonly used plots where the horizontal axis is the iteration counter. All experiments were stopped when the number of sample gradients evaluated reached the size of the complete training set. Note that for our choices of the growth rates, when $s_k = u_k$, roughly half of the available training points were seen by the algorithm. The last remaining parameter, α, was chosen to give the best testing function value at the end of the training epoch. A complete collection of these experiments for each of the six methods, on all four test problems can be seen in Figures 3.1, 3.3, 3.2, and 3.4. These extensive experiments are summarized in Figures 3.5, 3.7, 3.6, and 3.8 respectively. These summary plots show the best performing growth rates compared against each other.
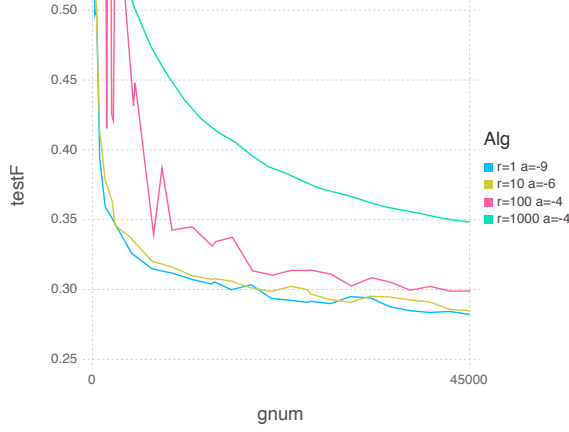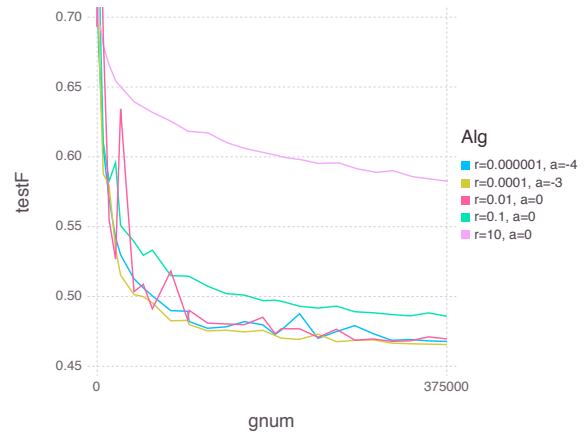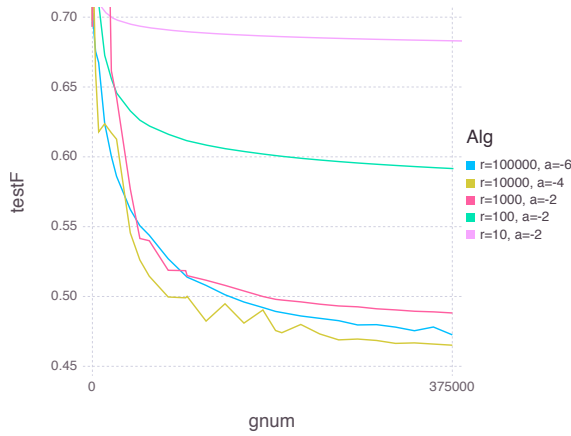
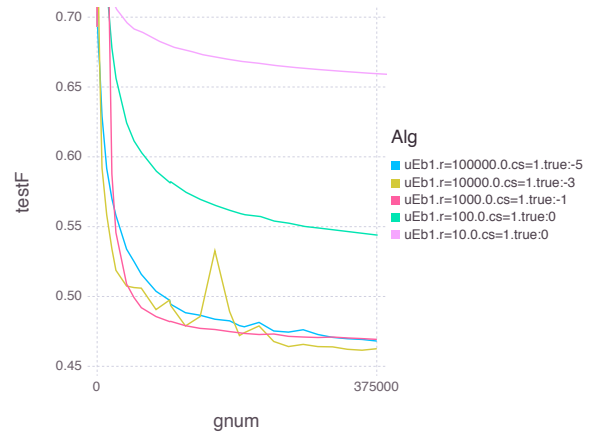(a) EGR-lin(SAG)
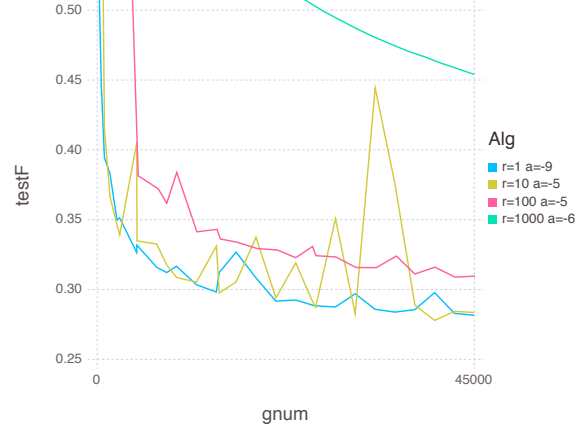
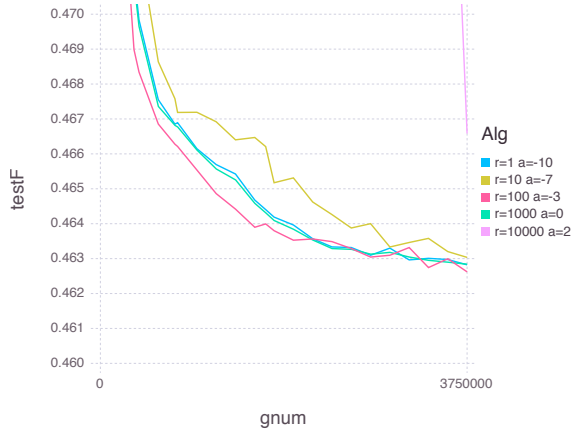(b) EGR-lin(SAGA)
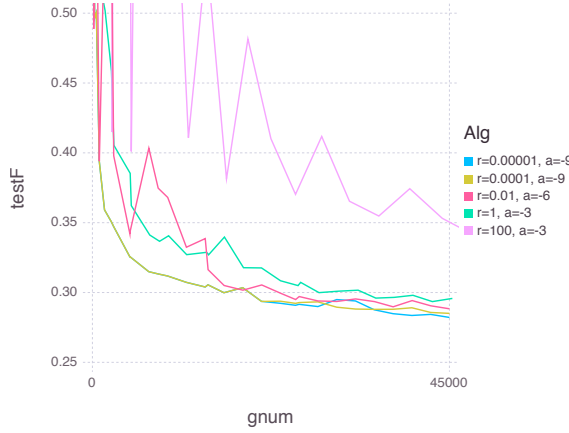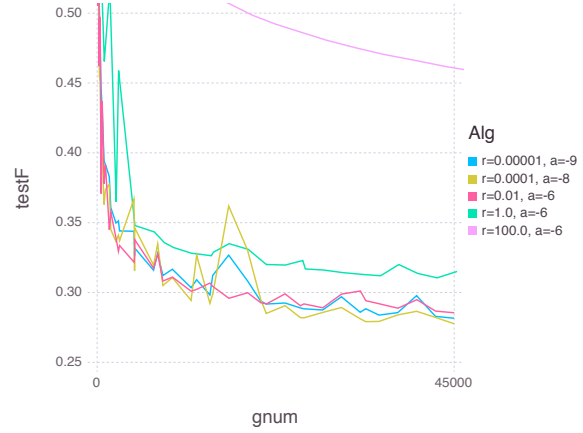
(c) EGR-quad(SAG)
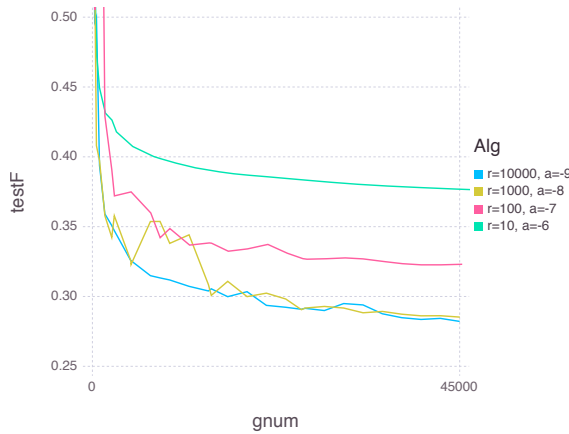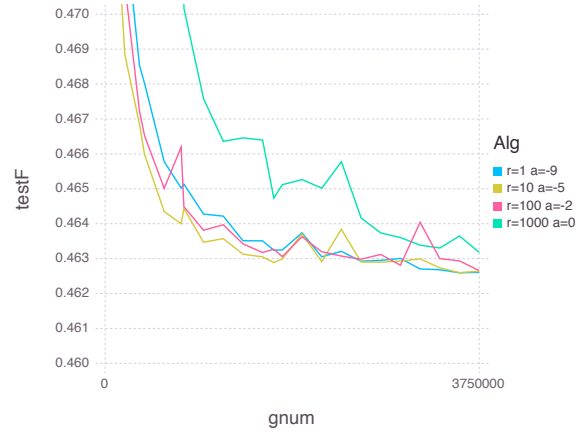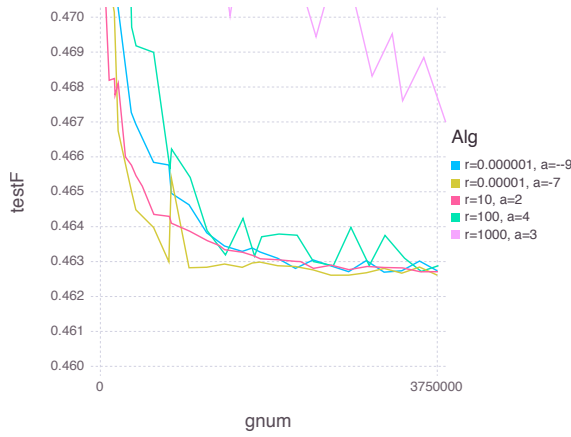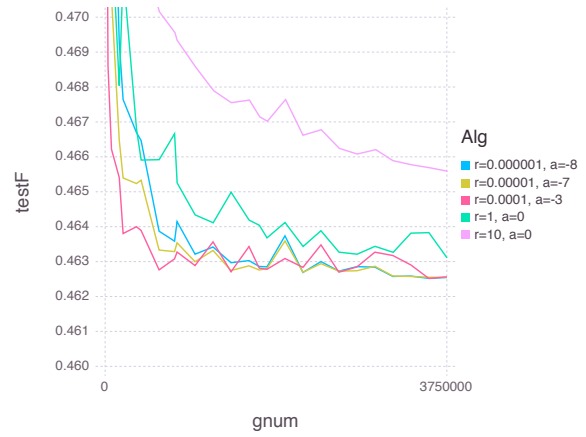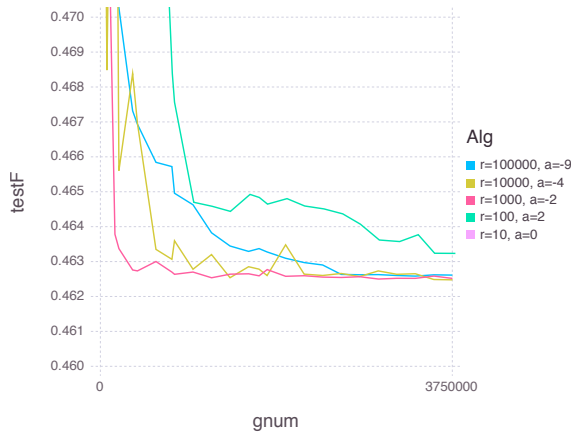
(d) EGR-quad(SAGA)
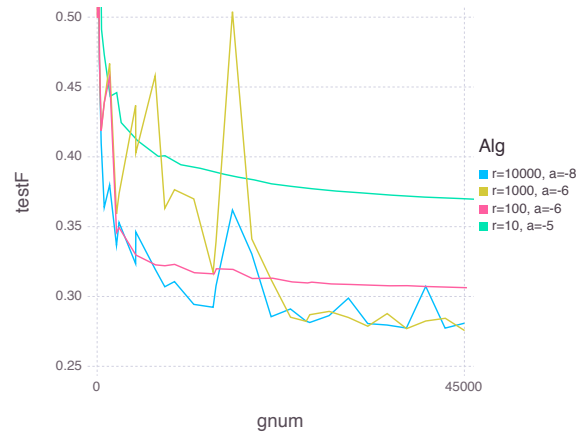
(e) EGR-exp(SAG)

(f) EGR-exp(SAGA)

Figure 3.1. MNIST detailed: Comparing various growth rates. In the legend, r is the parameter from Table 3.1 and $a = log_2(\alpha)$ where α is the constant stepsize parameter.
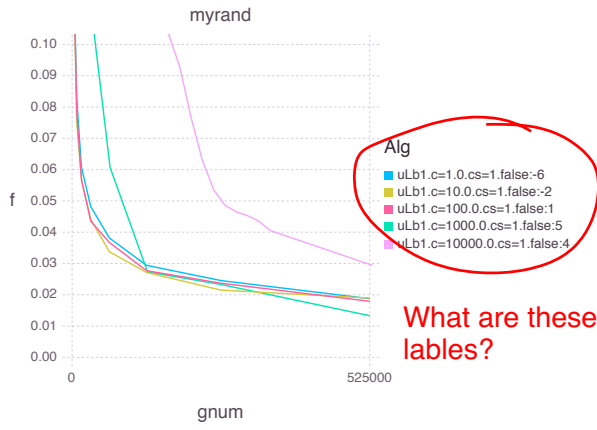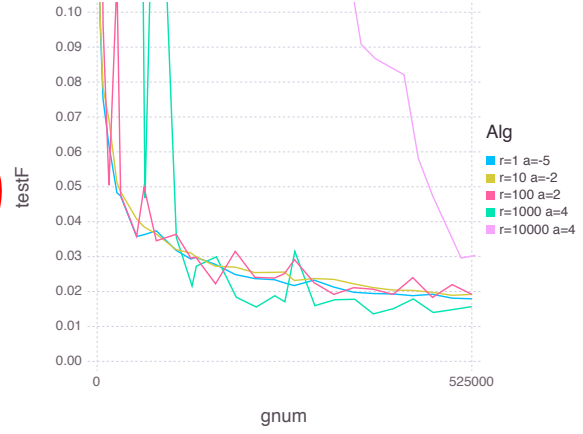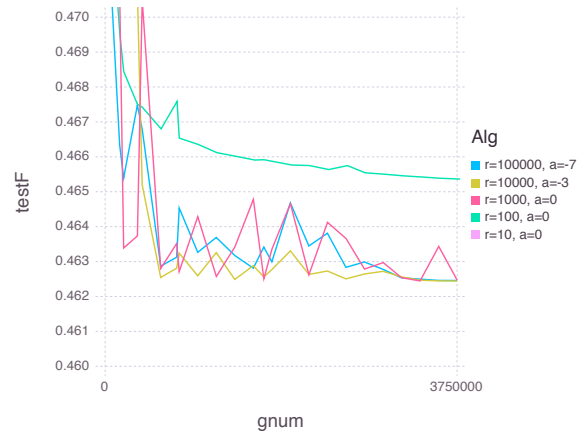
(a) EGR-lin(SAG)
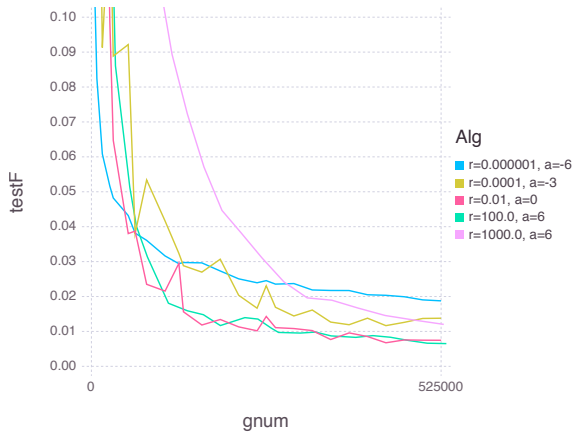
(b) EGR-lin(SAGA)
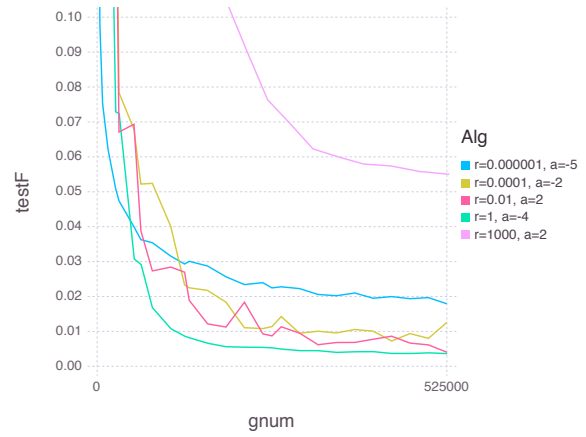
(c) EGR-quad(SAG)
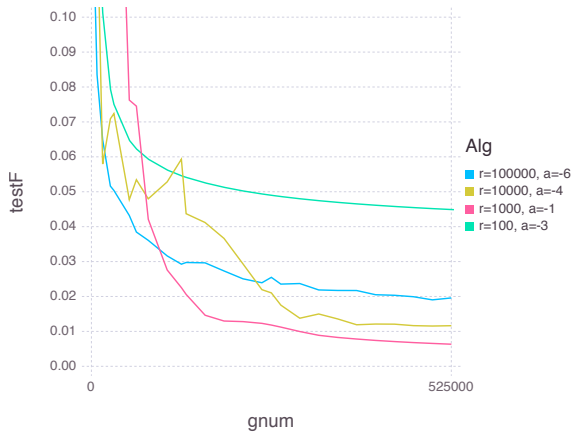
(d) EGR-quad(SAGA)
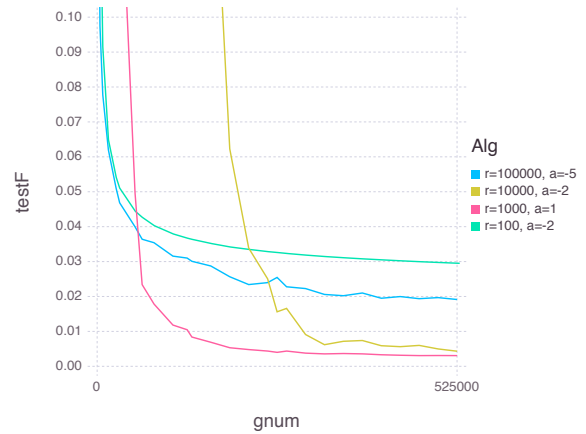
(e) EGR-exp(SAG)

(f) EGR-exp(SAGA)

Figure 3.2. myrand detailed: Comparing various growth rates. In the legend, r is the parameter from Table 3.1 and $a = log_2(\alpha)$ where α is the constant stepsize parameter.
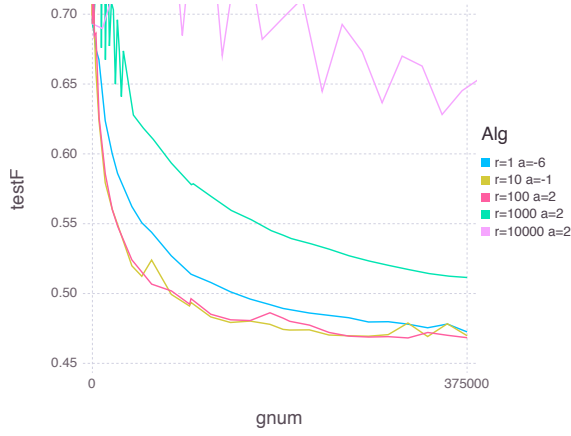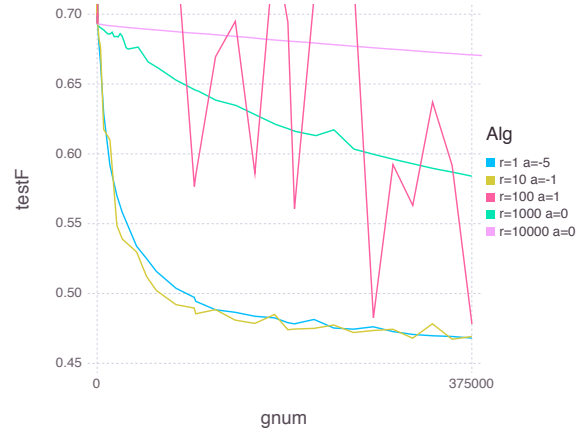
(a) EGR-lin(SAG)
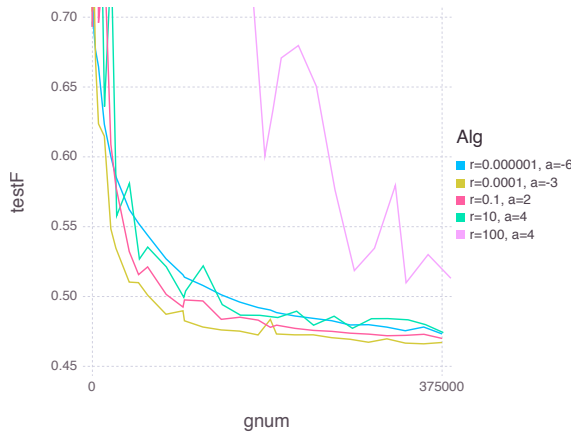
(b) EGR-lin(SAGA)

(c) EGR-quad(SAG)

(d) EGR-quad(SAGA)

(e) EGR-exp(SAG)

(f) EGR-exp(SAGA)

Figure 3.3. alphaGood detailed: Comparing various growth rates. In the legend, r is the parameter from Table 3.1 and $a = log_2(\alpha)$ where α is the constant stepsize parameter.

(a) EGR-lin(SAG)

(b) EGR-lin(SAGA)

(c) EGR-quad(SAG)

(d) EGR-quad(SAGA)

(e) EGR-exp(SAG)

(f) EGR-exp(SAGA)

Figure 3.4. SUSY detailed: Comparing various growth rates. In the legend, r is the parameter from Table 3.1 and $a = log_2(\alpha)$ where α is the constant stepsize parameter.
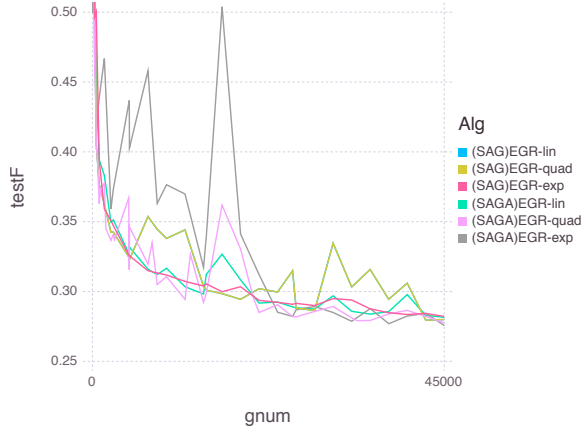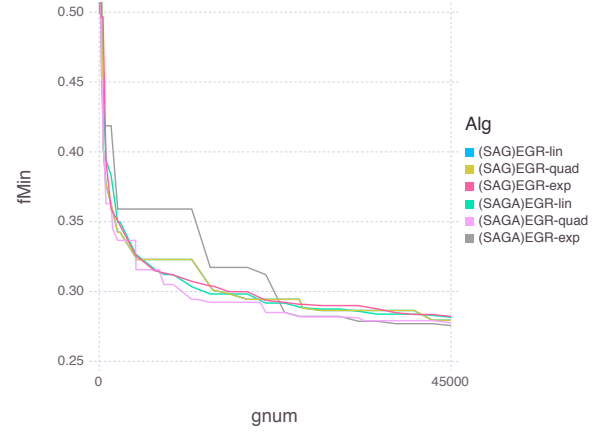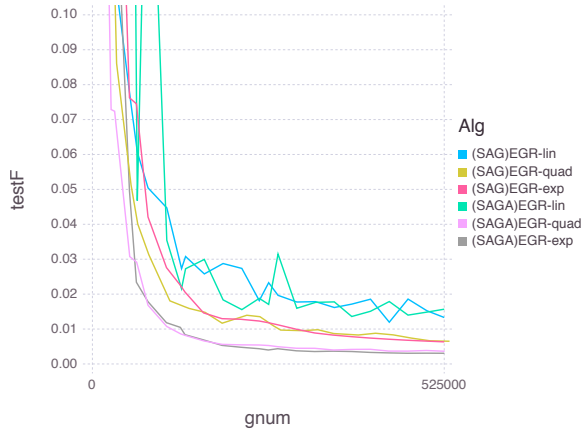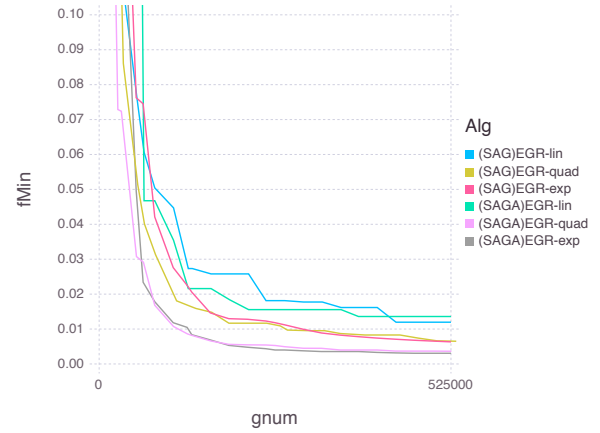
(a) Current F value   (b) Incumbent best solution
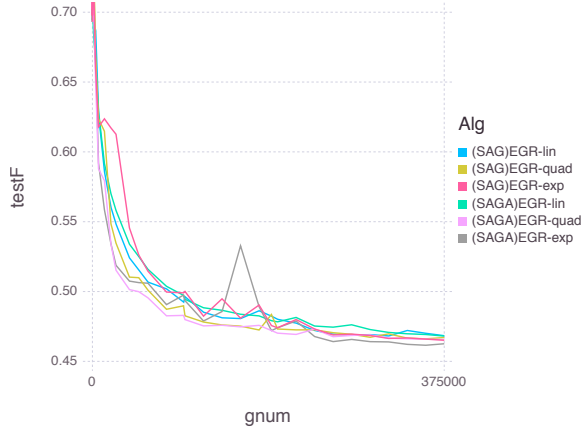
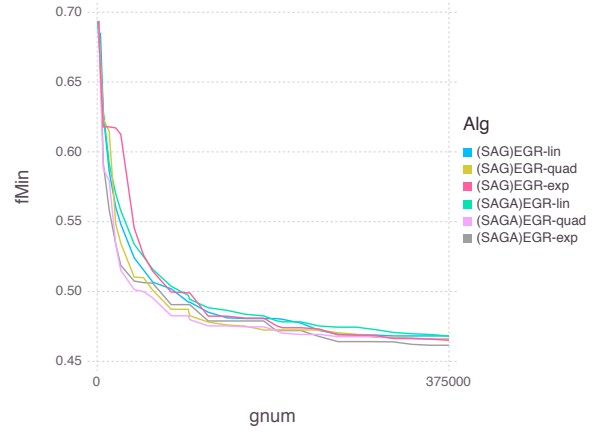Figure 3.5. MNIST summary: all EGR methods

(a) Current F value   (b) Incumbent best solution

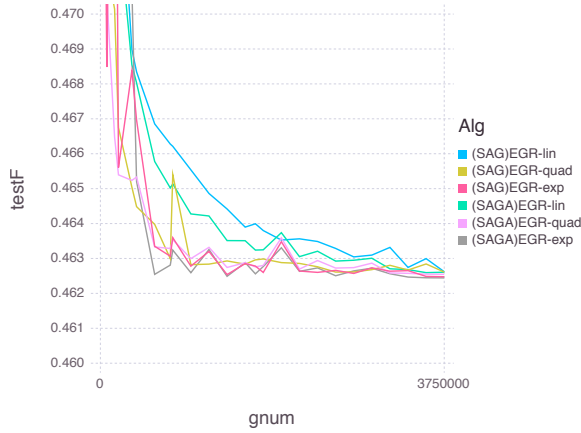Figure 3.6. myrand summary: all EGR methods
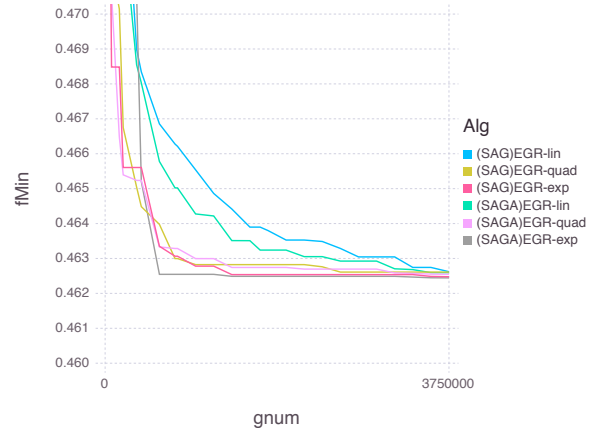
(a) Current F value

(b) Incumbent best solution

Figure 3.7. alphaGood summary: all EGR methods

(a) Current F value

(b) Incumbent best solution

Figure 3.8. SUSY summary: all EGR methods

On problem MNIST, in Figure 3.1, we see that for EGR-lin(SAG) smaller batch sizes are preferable. When $r = 1$, the method performs extremely well, but slightly increasing the batch to $r = 10$ allows for significantly larger stepsizes to be taken. This closely

follows the common knowledge about the need for mini-batching in SG methods. As r increases, larger steplengths are needed for good performance: this is logical since the methods use more new information at each iteration, thus having less variance in the step directions. Large batch sizes do not work well, since the method is only able to take a few iterations in the allotted sample gradient evaluation limit. The EGR-lin(SAGA) version shows similar patterns, but a higher variance in the algorithm progression is seen. Smaller stepsizes than in EGR-lin are needed: this is due to the fact that the SAG version of the algorithm places a much larger emphasis on the relatively stable term involving the summation of historic sample gradient information. In the quadratic EGR-SAG growth rate case, the smallest r tested gives the same method as the linear $r = 1$ method because of rounding. The second growth rate (corresponding to $r = 0.0001$) behaves identical to the smallest one initially, but approximately after the first half of the graph, the two plots diverge. This is because with the second growth rate the batch sizes start increasing from $s_k = u_k = 1$. In the quadratic EGR-SAGA case, this sudden increase of the sample sizes enables the superior performance of the method. For the exponentially growing case, the same idea as before applies: A slightly increasing batch size in the latter iterations seems to benefit the stability and sometimes the performance of the methods.

In Figure 3.5, we combine these methods in a two plots. The one on the left always shows the true progression of the algorithm to the minimizer, i.e. the current estimate of the function value of F. The plot on the right shows the same data plotted to display the current best solution. The EGR-SAGA methods with growing batch sizes at the end seem to work best in practice. We attribute this to the unbiased nature of the methods.

We see that the best performing methods are the ones which increase the sample sizes at the end, instead of leaving them at a constant level.

With the myrand dataset, the picture is slightly different. In Figure 3.2, we can see that the required growth rates are higher than for MNIST. In Figure 3.6, we again see the clear advantage of using growing batch sizes, and of SAGA step computation formula over the SAG one.

On the alphaGood dataset, and the SUSY dataset, the conclusions are similar. In Figure 3.9 we provide a plot of current sample size, $s_k + u_k$, at each iteration of the algorithm, on the MNIST dataset. The best performing growth rate choice seems to be the one where approximately for this first half of the algorithm progression the sample sizes are as small as possible, and only then start growing.
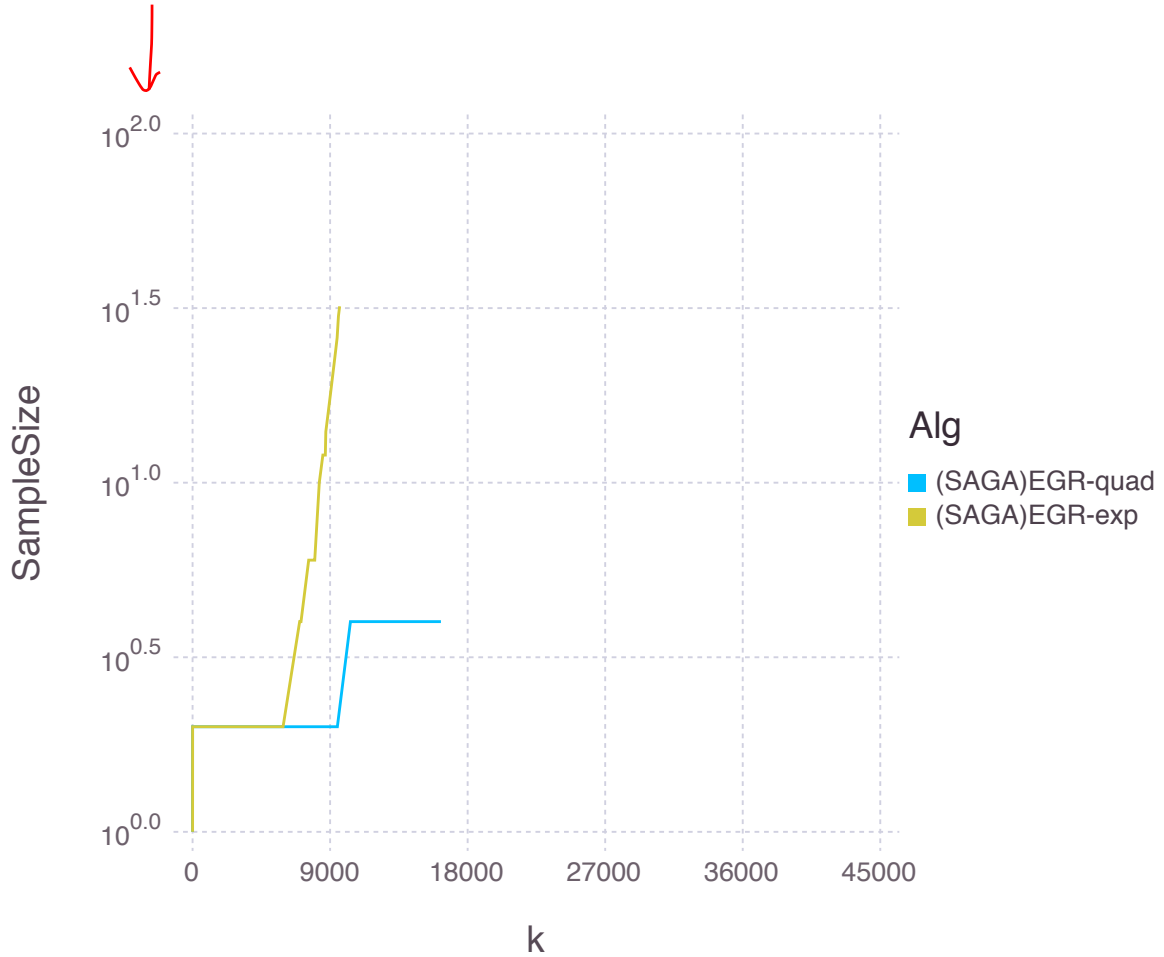
Use actual numbers here?

Figure 3.9. myrand: EGR vs other methods

3.4.2.2. Comparison with SG and Dynamic Sampling. The only-add option in EGR includes many existing methods, such as SG and DSS. We empirically compare the performance of our new methods with SG and DSS. The Stochastic Gradient method enjoys immense popularity in both the optimization and the machine learning communities, and when carefully tuned can outperform any known modern method. The tuning often involves intricate step size strategies. We do not attempt to compare with the best SG implementation, rather we demonstrate the benefits of using the EGR scheme, compared

to SG, with both methods using a constant step size. The DSS method tested uses a sample size identical to the one suggested by EGR, to make the per-iteration comparison fair. The plots for each problem can be found in Figures 3.10, 3.11, 3.12, 3.13.
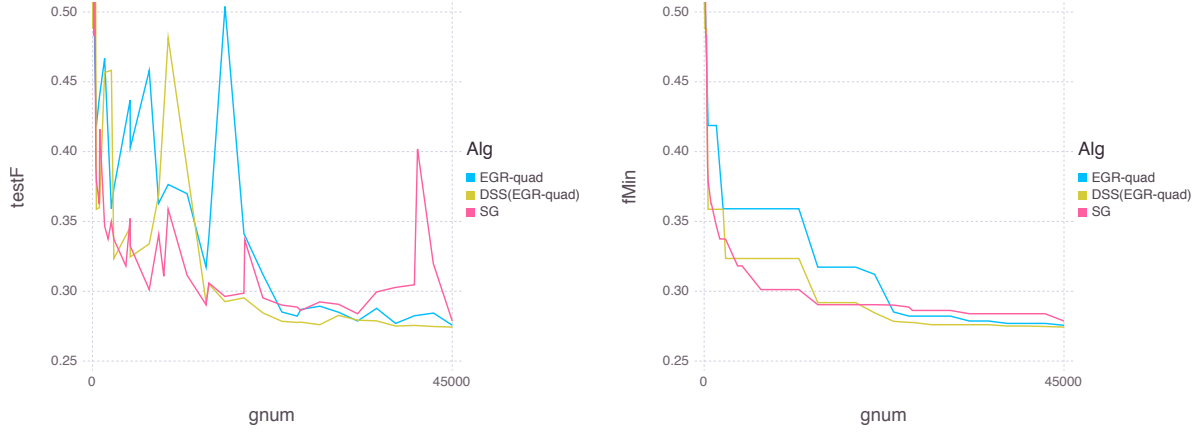
(a) Current F value

(b) Incumbent best solution

Figure 3.10. MNIST: EGR vs only-add methods
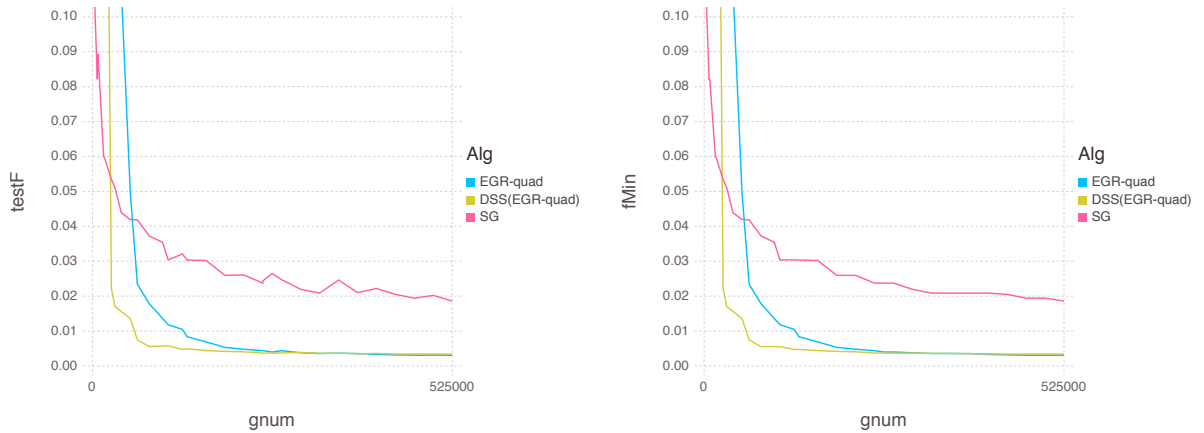
(a) Current F value

(b) Incumbent best solution

Figure 3.11. myrand: EGR vs only-add methods

(a) Current F value

(b) Incumbent best solution

Figure 3.12. alphaGood: EGR vs only-add methods

(a) Current F value
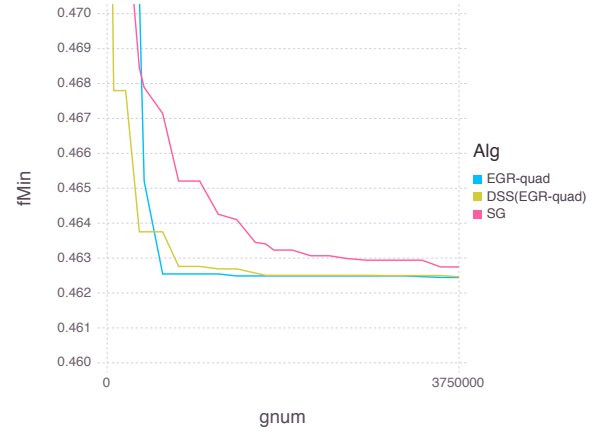
(b) Incumbent best solution

Figure 3.13. SUSY: EGR vs only-add methods

These plots show that the benefit of having multiple sample gradients evaluated at some iterations is advantageous to having a single one being evaluated at each step. The

advantage of ==using EGR over DSS== is not as pronounced: ==we cannot make a clear statement about the best performing method in this case.==

3.4.2.3. Comparison with SAG-init Methods. We compare the EGR method with two other similar algorithms: The SAGinit and SAGAinit methods are not commonly thought of as optimization methods, but are presented as initialization heuristics in the corresponding papers. Nevertheless, the methods are considered as a way to run these aggregated gradient methods while building up memory as you go along. The EGR framework includes algorithms which outperform these heuristics: they are able to consistently reach a better F value, see Figure 3.15.

(a) Current F value   (b) Incumbent best solution
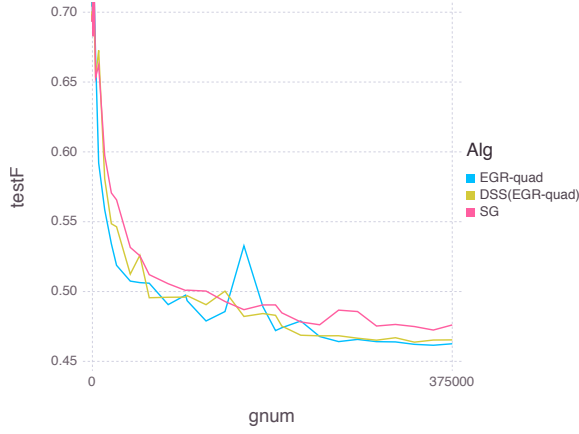
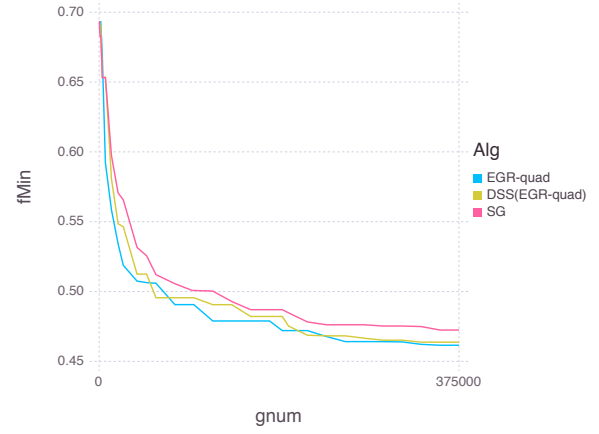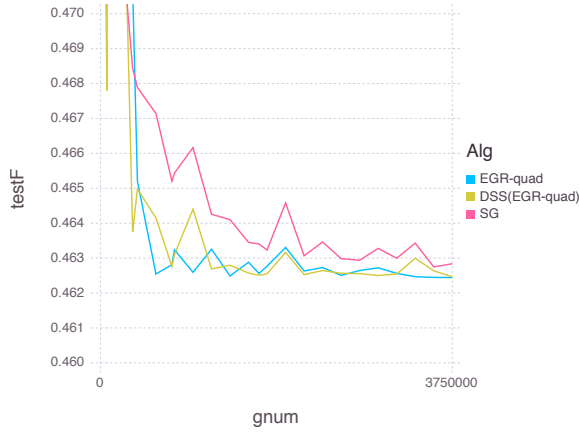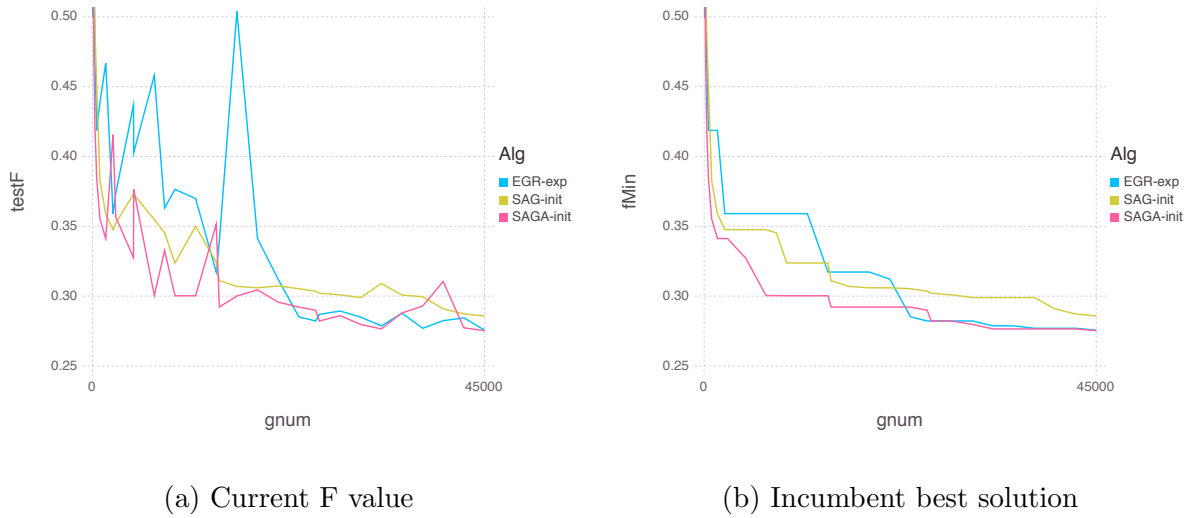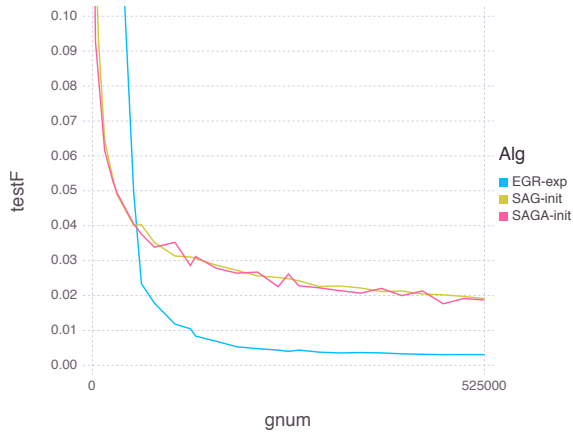Figure 3.14. MNIST: EGR vs other methods
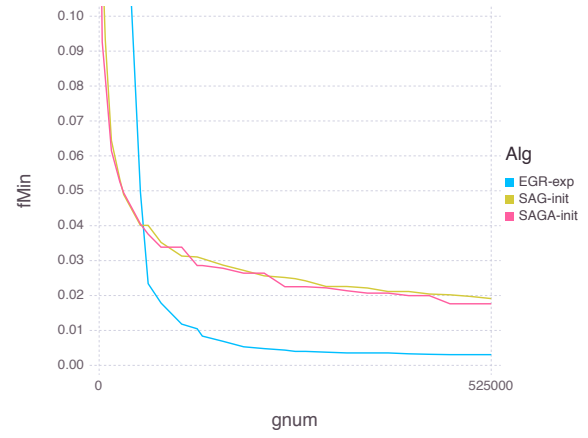
(a) Current F value

(b) Incumbent best solution

Figure 3.15. myrand: EGR vs other methods
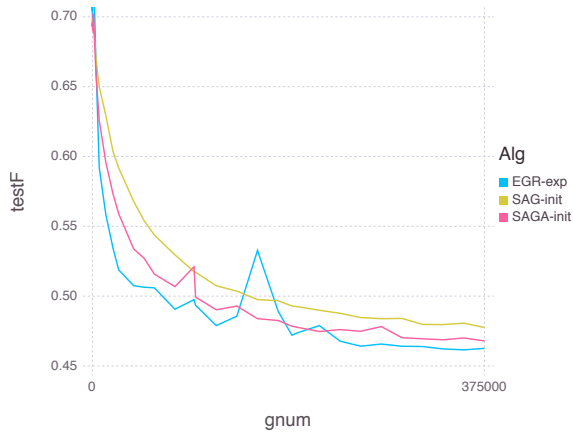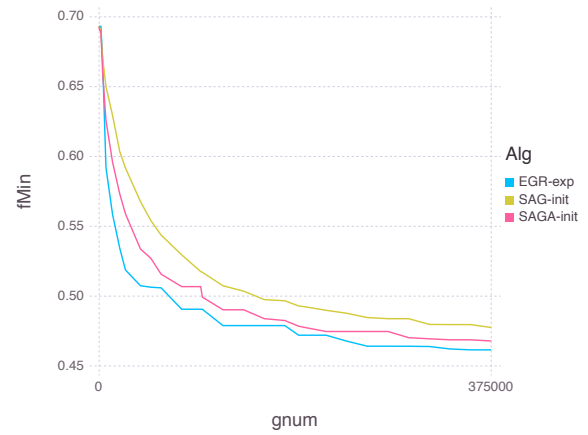
(a) Current F value

(b) Incumbent best solution

Figure 3.16. alphaGood: EGR vs other methods

(a) Current F value

(b) Incumbent best solution

Figure 3.17. SUSY: EGR vs other methods

3.5. Final Remarks

We have proposed a stochastic optimization method that has the ability to re-use gradient information as well as add new gradients at each iteration. The generality of our framework is appealing as it includes the stochastic gradient method, SAG, and dynamic sampling methods, and also gives rise to new algorithms.

We have established (global) linear convergence of the method for strongly convex functions. Our analysis assumes that the sample grows geometrically, which is an efficient choice in practice. Our proof technique differs from the techniques use to establish convergence of SAG, SAGA and dynamic sampling methods. We employ a novel error function (in lieu of a Lyapunov function) that makes the analysis more compact and provides, as a side product, a short proof for SAG.

Our numerical experiments indicate that the method proposed in this thesis is efficient in a variety of test problems. Our overall conclusion is that re-using gradient information

in the context of a dynamic sampling approach results in a stable and competitive algorithm. Much more extensive testing, which is outside of the scope of this thesis, will be required to gauge the full potential of our method.

References

[1] Afonso, M., Bioucas-Dias, J., and Figueiredo, M. A. T. Fast image recovery using variable splitting and constrained optimization. *Image Processing, IEEE Transactions on 19*, 9 (2010), 2345–2356.

[2] Andrew, G., and Gao, J. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine Learning* (2007), ACM, pp. 33–40.

[3] Beck, A., and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences 2*, 1 (2009), 183–202.

[4] Becker, S. R., Candés, E. J., and Grant, M. C. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation 3*, 3 (2011), 165–218.

[5] Bertsekas, D. P. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning* (2010), 1–38.

[6] Bioucas-Dias, J., and Figueiredo, M. A new TwIST: two-step iterative Shrinkage/Thresholding algorithms for image restoration. *IEEE Transactions on Image Processing 16*, 12 (Dec. 2007), 2992–3004.

[7] Byrd, R., Nocedal, J., and Oztoprak, F. An inexact successive quadratic approximation method for convex l1 regularized optimization. *arXiv preprint arXiv:1309.3529* (2013). To appear in *Mathematical Programming B*.

[8] Byrd, R. H., Chin, G. M., Nocedal, J., and Oztoprak, F. A family of second-order methods for convex L1 regularized optimization. Tech. rep., Optimization Center Report 2012/2, Northwestern University, 2012.

[9] Byrd, R. H., Chin, G. M., Nocedal, J., and Wu, Y. Sample size selection in optimization methods for machine learning. *Mathematical Programming 134*, 1 (2012), 127–155.

[10] CHANG, C.-C., AND LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[11] DAUBECHIES, I., DEFRISE, M., AND DE MOL, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics 57*, 11 (2004), 1413–1457.

[12] DEFAZIO, A., BACH, F., AND LACOSTE-JULIEN, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1646–1654.

[13] DEFAZIO, A. J., CAETANO, T. S., AND DOMKE, J. Finito: A faster, permutable incremental gradient method for big data problems. *arXiv preprint arXiv:1407.2710* (2014).

[14] DENG, W., YIN, W., AND ZHANG, Y. Group sparse optimization by alternating direction method. In *SPIE Optical Engineering+ Applications* (2013), International Society for Optics and Photonics, pp. 88580R–88580R.

[15] DOLAN, E. D., AND MORÉ, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A 91* (2002), 201–213.

[16] DONOHO, D. De-noising by soft-thresholding. *Information Theory, IEEE Transactions on 41*, 3 (1995), 613–627.

[17] DOSTAL, Z., AND SCHOEBERL, J. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Computational Optimization and Applications 30*, 1 (2005), 23–43.

[18] DUCHI, J., HAZAN, E., AND SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research 999999* (2011), 2121–2159.

[19] EFRON, B., HASTIE, T., JOHNSTONE, I., AND TIBSHIRANI, R. Least angle regression. *The Annals of statistics 32*, 2 (2004).

[20] FIGUEIREDO, M., NOWAK, R., AND WRIGHT, S. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of 1*, 4 (dec. 2007), 586 –597.

[21] FOUNTOULAKIS, K., AND GONDZIO:, J. A second-order method for strongly-convex l1-regularization problems. *Technical Report ERGO-14-005* (2014).

[22] FRIEDLANDER, M., AND SCHMIDT, M. Hybrid deterministic-stochastic methods for data fitting. *Arxiv preprint arXiv:1104.2373* (2011).

[23] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software 33*, 1 (2010), 1.

[24] FROSTIG, R., GE, R., KAKADE, S. M., AND SIDFORD, A. Competing with the empirical risk minimizer in a single pass. *arXiv preprint arXiv:1412.6606* (2014).

[25] FUCHS, J. J. More on sparse representations in arbitrary bases. *IEEE Trans. on I.T* (2004), 1341–1344.

[26] GAFNI, E. M., AND BERTSEKAS, D. P. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization 22*, 6 (1984), 936–964.

[27] HALE, E. T., YIN, W., AND ZHANG, Y. Fixed-point continuation for \ell_1-minimization: Methodology and convergence. *SIAM Journal on Optimization 19*, 3 (2008), 1107–1130.

[28] HANSEN, S., AND NOCEDAL, J. Second-order methods for L_1 regularized problems in machine learning. Tech. rep., Optimization Center Report 2011/3, Northwestern University, 2011. To appear in ICAASP 2012.

[29] HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations.* New York: Springer-Verlag, 2001.

[30] JOHNSON, R., AND ZHANG, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems* (2013), pp. 315–323.

[31] KALIVAS, J. H. Two data sets of near infrared spectra. *Chemometrics and Intelligent Laboratory Systems 37*, 2 (June 1997), 255–259.

[32] KIM, S.-J., KOH, K., LUSTIG, M., BOYD, S., AND GORINEVSKY, D. An interior-point method for large-scale l1-control-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing 1*, 4 (Dec. 2007), 606–617.

[33] KIM, S.-J., KOH, K., LUSTIG, M., BOYD, S., AND GORINEVSKY, D. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of 1*, 4 (2007), 606–617.

[34] MAIRAL, J. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization 25*, 2 (2015), 829–855.

[35] MILZAREK, A., AND ULBRICH, M. A semismooth Newton method with multi-dimensional filter globalization for L1-optimization. *SIAM Journal on Optimization 24*, 1 (2014).

[36] NESTEROV, Y. A method for solving the convex programming problem with convergence rate o(1/k2). *Dokl. Akad. Nauk SSSR 269* (1983), 543–547.

[37] NESTEROV, Y. *Introductory Lectures on Convex Optimization: A Basic Course.* Boston: Kluwer Academic Publishers, 2004.

[38] NESTEROV, Y. Primal-dual subgradient methods for convex problems. *Math. Program. 120*, 1 (2009), 221–259.

[39] NOCEDAL, J., AND WRIGHT, S. *Numerical Optimization*, 2 ed. Springer New York, 1999.

[40] OLSEN, P., OZTOPRAK, F., NOCEDAL, J., AND RENNIE, S. Newton-like methods for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. 2012, pp. 764–772.

[41] PASUPATHY, R., GLYNN, P. W., GHOSH, S., AND HAHEMI, F. How much to sample in simulation-based stochastic recursions? Under Review.

[42] PERKINS, S., LACKER, K., AND THEILER, J. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research 3* (2003), 1333–1356.

[43] POLYAK, B., AND JUDITSKY, A. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization 30* (1992), 838.

[44] ROBBINS, H., AND MONRO, S. A stochastic approximation method. *The Annals of Mathematical Statistics* (1951), 400–407.

[45] Roux, N. L., Schmidt, M., and Bach, F. R. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems* (2012), pp. 2663–2671.

[46] Ruppert, D. Efficient estimations from a slowly convergent robbins-monro process. Tech. rep., Cornell University Operations Research and Industrial Engineering, 1988.

[47] Schmidt, M. *Graphical Model Structure Learning with L1-Regularization.* PhD thesis, University of British Columbia, 2010.

[48] Schmidt, M., Fung, G., and Rosales, R. Fast optimization methods for l1 regularization: A comparative study and two new approaches suplemental material.

[49] Shalev-Shwartz, S., and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research 14*, 1 (2013), 567–599.

[50] Sra, S., Nowozin, S., and Wright, S. *Optimization for Machine Learning.* Mit Press, 2011.

[51] Wainwright, M. Sharp thresholds for high-dimensional and noisy sparsity recovery using -constrained quadratic programming (lasso). *Information Theory, IEEE Transactions on 55*, 5 (May), 2183–2202.

[52] Wen, Z., Yin, W., Goldfarb, D., and Zhang, Y. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation. *SIAM Journal on Scientific Computing 32*, 4 (2010), 1832–1857.

[53] Wen, Z., Yin, W., Goldfarb, D., and Zhang, Y. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM J. Sci. Comput. 32*, 4 (June 2010), 1832–1857.

[54] Wright, S., Nowak, R., and Figueiredo, M. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing 57*, 7 (2009), 2479–2493.

[55] Zou, H., and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*, 2 (2005), 301–320.

Appendices

.1. Dataset Details and Sparsity Patterns

The τ values for each problem were chosen by experimentation so as to span a range of solution sparsities. This is preferable to setting τ as a multiple of $\|b\|_\infty$ (as is often done in the literature based on the fact when $\tau = \|b\|_\infty$ the optimal solution to problem (2.1) is the zero vector [**25**]). We prefer to select the value of τ for each problem, as there sometimes is a very small range of values that yields interesting problems.

.2. Effect of overestimating $\|A\|$ in the Gradient Balance Condition

In our experiments, we set $\alpha = 1/L$ in iiCG, in the gradient balance condition computation. Often L is not known and is hard to compute (for medium and large-scale

Table .3. `randQuad` $n = 2000$

$norm(A)$	$cond(A)$	γ	problem	τ	num zeros
5781.5	-	0	`randQuads1`	100	1001
			`randQuads2`	1000	1011
			`randQuads3`	10000	1133
			`randQuads4`	100000	1850
5781.5	5.7815e+06	0.001	`randQuadi1`	0.1	583
			`randQuadi2`	100	1011
			`randQuadi3`	10000	1133
			`randQuadi4`	100000	1850
5782.5	5782.5	1	`randQuadm1`	0.1	4
			`randQuadm2`	100	620
			`randQuadm3`	10000	1130
			`randQuadm4`	100000	1850

Table .4. `spectra` $n = 402$

$norm(A)$	$cond(A)$	γ	problem	τ	num zeros
2.056413e+03	-	0	spectras1	1.0e-06	322
			spectras2	1.0e-04	348
			spectras3	1.0e-03	372
			spectras4	1.0e-02	389
2.056414e+03	2.056414e+06	1.0e-03	spectrai1	3.0e-05	2
			spectrai2	1.0e-03	91
			spectrai3	1.0e-02	313
			spectrai4	5.0e-01	398
2.057413e+03	2.057413e+03	1	spectram1	1.0e-03	1
			spectram2	2.0e-01	109
			spectram3	1	332
			spectram4	30	388

Table .5. `sigrec` $n = 4096$

$norm(A)$	$cond(A)$	γ	problem	τ	num zeros
1.119904e+00	-	0	sigrecs1	5.0e-05	3549
			sigrecs2	2.0e-04	3816
			sigrecs3	5.0e-03	3860
			sigrecs4	1.0e-01	3973
1.119905e+00	1.119905e+06	1.0e-06	sigreci1	5.0e-08	828
			sigreci2	5.0e-05	3535
			sigreci3	2.0e-04	3813
			sigreci4	1.0e-01	3973
1.120904e+00	1.120904e+03	1.0e-03	sigrecm1	4.5e-07	16
			sigrecm2	1.0e-04	1519
			sigrecm3	2.0e-03	3310
			sigrecm4	1.0e-01	3973

problems computing L may take longer than running the algorithm itself). Figure .18 shows that while $\alpha = \frac{1}{L}$ is a good choice, iiCG-2 is fairly insensitive to the choice of α, particularly if the value $1/L$ is overestimated.

Table .6. `proxnewt` $n = 5000$

| $norm(A)$	$cond(A)$	γ	problem	τ	num zeros
			`proxnewts1`	6.7e-06	1893
1.103666e+02	-	0	`proxnewts2`	6.7e-05	3192
			`proxnewts3`	6.7e-04	4365
			`proxnewts4`	6.7e-03	4960
			`proxnewti1`	6.7e-06	1395
1.103667e+02	1.103667e+06	1.0e-04	`proxnewti2`	6.7e-05	3060
			`proxnewti3`	6.7e-04	4344
			`proxnewti4`	6.7e-03	4959
			`proxnewtm1`	6.7e-06	193
1.103771e+02	1.051211e+04	1.0e-02	`proxnewtm2`	6.7e-05	1283
			`proxnewtm3`	6.7e-04	3602
			`proxnewtm4`	6.7e-03	4926
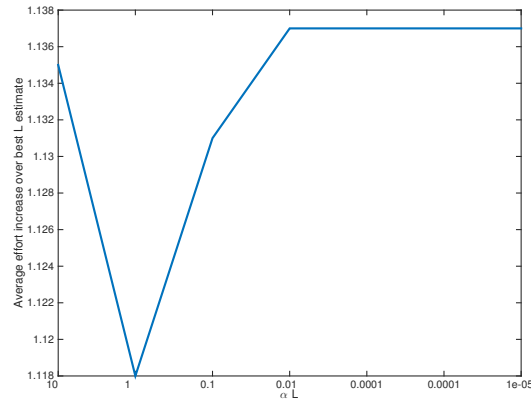
Figure .18. Average increase in matrix-vector products relative to the optimal choice for α (obtained by experimentation), for various choices of α. The results are compiled from all 48 test problems, and the runs were stopped when `tol`$=10^{-4}$.