A Superlinearly-Convergent Proximal Newton-Type Method for the Optimization of Finite Sums

Anton Rodomanov ANTON.RODOMANOV@GMAIL.COM
Higher School of Economics, Faculty of Computer Science, 125319, Kochnovsky Proezd, 3, Moscow, RUSSIA

Dmitry Kropotov DMITRY.KROPOTOV@GMAIL.COM
Lomonosov Moscow State University, 119991, CMC Faculty, Leninskie Gory, 1, Moscow, RUSSIA

Abstract

We consider the problem of optimizing the strongly convex sum of a finite number of convex functions. Standard algorithms for solving this problem in the class of incremental/stochastic methods have at most a linear convergence rate. We propose a new incremental method whose convergence rate is superlinear – the Newton-type incremental method (NIM). The idea of the method is to introduce a model of the objective with the same sum-of-functions structure and further update a single component of the model per iteration. We prove that NIM has a superlinear local convergence rate and linear global convergence rate. Experiments show that the method is very effective for problems with a large number of functions and a small number of variables.

1. Introduction

In this paper we consider the following strongly convex unconstrained optimization problem:

$$\min_{x\in\mathbf{R}^d}\left[\phi(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x)\right], \qquad (1)$$

where all $f_i : \mathbf{R}^d \to \mathbf{R}$ are twice continuously differentiable convex functions and $h : \mathbf{R}^d \to \mathbf{R}$ is convex, but not necessarily differentiable. We also assume that for function $h(x)$ we have access to some efficient implementation of its proximal mapping

$$\operatorname{prox}_h(x) := \operatorname*{argmin}_{y\in\mathbf{R}^d}\left[h(y) + \frac{1}{2}\|y-x\|^2\right]. \qquad (2)$$

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

A typical example of problem (1) is regularized empirical risk minimisation for training a machine learning algorithm. In this case the variables x are the parameters of the model, $f_i(x)$ measures the error of the model on the ith training sample and $h(x)$ is a regularizer, e.g. l_1- or l_1/l_2-norm. Since the objective ϕ is supposed to be strongly convex, it has the unique minimizer which we denote by x^*.

We consider the case when the number of functions n may be very large. In this situation for minimizing ϕ it is convenient to use incremental optimization methods (Bertsekas, 2011) since the complexity of their iteration does not depend on n. Unlike classical optimization methods which process all the n functions at every iteration, incremental methods in each iteration work only with a single component f_i[1]. If each incremental iteration tends to make reasonable progress in some "average" sense, then an incremental method may significantly outperform its non-incremental counterpart (Bertsekas, 2011).

There is a large body of work in the field of incremental optimization methods (Byrd et al., 2014; Bordes et al., 2009; Schraudolph et al., 2007). They all can be divided into two groups depending on their rate of convergence.

The first group contains the stochastic gradient method (SGD) and other methods with iterations of the form, $x_{k+1} = \operatorname{prox}_h(x_k - \alpha_k B_k \nabla f_{i_k}(x_k))$, where B_k is some scaling matrix. For instance, in SGD B_k is just the identity matrix; in methods oBFGS and oLBFGS (Schraudolph et al., 2007), the matrix B_k is set to a quasi-Newton estimate of the inverse Hessian; in method SGD-QN (Bordes et al., 2009) the matrix B_k is a diagonal matrix whose entries are estimated from the secant equation; method SQN (Byrd et al., 2014) is an advanced version of oLBFGS; instead of estimating the gradient difference $\nabla f(x_{k+1}) - \nabla f(x_k)$ by subtracting two stochastic gradi-

[1]Usually all the functions f_i are grouped in small subsets called mini-batches with further processing of one mini-batch per incremental iteration

ents, SQN estimates it by multiplying a stochastic Hessian by the deterministic difference $x_{k+1} - x_k$. Compared to SGD, all these methods may have better practical performance. However, none of them qualitatively improves on the rate of convergence—any method from this group has a slow sublinear rate of convergence, including the method with $B_k = \nabla^2 f(x_k)^{-1}$ (Bousquet & Bottou, 2008). Although not all of these methods support proximal mapping.

The second group contains methods such as SAG (Schmidt et al., 2013), IAG (Blatt et al., 2007), SVRG (Johnson & Zhang, 2013), FINITO (Defazio et al., 2014b), SAGA (Defazio et al., 2014a), MISO (Mairal, 2015) etc. The common property of these methods is that they use such estimates of the objective gradient $\nabla f(x_k)$ whose error tends to zero as the iterate x_k approaches the optimum x^*. As a result, these methods may converge with constant step lengths, and their rate of convergence is linear.

Several recent works (Kolte et al., 2015; Lucchi et al., 2015; Moritz et al., 2015) try to incorporate a second-order information into linear-convergent incremental methods, especially SVRG. These methods are more robust to choice of parameters and may give better results for poorly conditioned problems. However, for all of them the rate of convergence remains linear.

The goal of this paper is to introduce an incremental optimization method with a fast *superlinear* local rate of convergence—the Newton-type incremental method (NIM). To the best of our knowledge, this is the first method in the class of incremental methods whose rate of convergence is superlinear. The idea of the method is to consider a model of the objective with the same sum-of-functions structure and further update a single component of the model per iteration. We provide a theoretical analysis of the convergence rates of NIM, both local and global. Besides, we introduce and analyze inexact version of NIM in spirit of inexact Newton method and inexact proximal Newton method (Lee et al., 2012).

In the recent work (Gürbüzbalaban et al., 2015) the incremental Newton (IN) method was proposed. Despite the similar name, the IN method is completely different from NIM. This method belongs to the first group of incremental methods with B_k equal to a partial sum of $\nabla^2 f_i$. The rate of convergence for IN is sublinear.

The most closely-related method to NIM is SFO (Sohl-Dickstein et al., 2014). This method also uses a second-order model for each function f_i, but, instead of the true Hessian $\nabla^2 f_i(v_i^k)$, it works with its approximation obtained with the aid of an L-BFGS-like technique. Although no convergence analysis of SFO is given in (Sohl-Dickstein et al., 2014), experiments show that SFO has a linear rate of convergence. This shows that the Hessian approximation of SFO is not accurate enough to ensure a superlinear rate. Besides, SFO does not support a proximal mapping.

The paper is organized as follows. First we introduce the method NIM for the general case of the strongly convex objective ϕ with arbitrary convex twice differentiable functions f_i. Also we consider an inexact version of the method and discuss in particular a new stopping criterion for the inner optimization problem that is eligible for incremental setting. Then in section 3 we make theoretical analysis of the proposed method both for local and global convergence. For the sake of clarity all technical proofs are moved to supplementary material. The next section considers the special case of linear models where method NIM has sufficiently less memory and iteration costs. We conclude with experiments and some discussion.

2. Method NIM

2.1. Model of the Objective

We begin the derivation of NIM by constructing a model of the objective ϕ at the current iteration k. First, we form the following convex quadratic model of each f_i:

$$f_i(x) \approx m_k^i(x) := f_i(v_k^i) + \nabla f_i(v_k^i)^\top (x - v_k^i) + \frac{1}{2}(x - v_k^i)^\top \nabla^2 f_i(v_k^i)(x - v_k^i)$$

for some point $v_k^i \in \mathbf{R}^d$. Once we have a model of each f_i, we can naturally build a model of the objective ϕ:

$$\phi(x) \approx m_k(x) := \frac{1}{n}\sum_{i=1}^n m_k^i(x) + h(x). \qquad (3)$$

In what follows we assume that the model (3) is strongly convex and thus has a unique minimum

$$\bar{x}_k = \operatorname*{argmin}_{x \in \mathbf{R}^d} m_k(x). \qquad (4)$$

To obtain the new iterate x_{k+1}, NIM first minimizes the current model (3) and then makes a step in the direction of the found minimum:

$$x_{k+1} := \alpha_k \bar{x}_k + (1 - \alpha_k) x_k,$$

where $\alpha_k \in (0, 1]$ is the step length. After the step is done, NIM updates the model m or, equivalently, the centers v. To keep the iteration complexity low, only one component of the full model is updated at every iteration:

$$v_{k+1}^i := \begin{cases} x_{k+1} & \text{if } i = i_k, \\ v_k^i & \text{otherwise,} \end{cases}$$

where $i_k \in \{1, \ldots, n\}$ is the index of the component to update.

In what follows we will use the notion of scaled proximal mapping:

$$\operatorname{prox}_h^H(x) = \operatorname*{argmin}_{y\in\mathbf{R}^d}\left[h(y) + \frac{1}{2}\|y-x\|_H^2\right], \qquad (5)$$

where H is some positive definite matrix and $\|x\|_H^2 := x^\top Hx$. Using simple rearrangement the problem (4) for the model (3) can be equivalently written as follows:

$$\bar{x}_k = \operatorname{prox}_h^{H_k}(H_k^{-1}(u_k - g_k)),$$

where we use the notation

$$u_k := \frac{1}{n}\sum_{i=1}^n \nabla^2 f_i(v_k^i)v_k^i, \quad g_k := \frac{1}{n}\sum_{i=1}^n \nabla f_i(v_k^i), \quad (6)$$
$$H_k := \frac{1}{n}\sum_{i=1}^n \nabla^2 f_i(v_k^i).$$

Note that in case $n = 1$ NIM becomes equivalent to proximal Newton method (Lee et al., 2012).

2.2. Model Update

Since we update only one component of the full model m at every iteration we need not compute the sums in (6) every time. Instead, we keep track of the quantities H_k, u_k, g_k and update them as follows (here i denotes the index of the selected component at iteration k):

$$H_{k+1} = H_k + \frac{1}{n}\left[\nabla^2 f_i(v_{k+1}^i) - \nabla^2 f_i(v_k^i)\right], \qquad (7)$$
$$u_{k+1} = u_k + \frac{1}{n}\left[\nabla^2 f_i(v_{k+1}^i)v_{k+1}^i - \nabla^2 f_i(v_k^i)v_k^i\right],$$
$$g_{k+1} = g_k + \frac{1}{n}\left[\nabla f_i(v_{k+1}^i) - \nabla f_i(v_k^i)\right].$$

In order to do this, we need to store all the centers v_k^i in memory. Taking into account the cost of storing H_k, u_k, g_k, the total storage cost of NIM is $O(nd + d^2)$. Note that we do not store the separate Hessians $\nabla^2 f_i(v_k^i)$ in memory because it would cost $O(nd^2)$. Therefore, to update the model, we need to compute the selected f_i twice—once at the new point x_{k+1} and once at the previous point v_k^i. Below (section 4.3) we analyze the special case of the objective ϕ for linear models. In this case memory costs of NIM can be sufficiently reduced.

2.3. Inexact Model Minimization

Unfortunately, even if the proximal mapping (2) is given in analytic form or can be efficiently estimated, the result of the scaled proximal mapping (5) is much harder to compute. Here we need to use a separate iterative optimization routine. Fortunately, there is a bunch of methods particularly tailored to the problem (5). Some of them are intended for specific functions $h(x)$, e.g. `glmnet` (Friedman et al., 2010) and `OWL-QN` (Andrew & Gao, 2007). Others can be applied for any convex $h(x)$ with efficient proximal mapping (2), e.g. `Fast gradient method (FGM)` (Nesterov, 2013). In what follows we will always use `Conjugate Gradient Method` for solving (5) in case $h(x)$ is l_2-norm and `FGM` in all other cases. We will refer any iterative method for solving subproblem (5) as `IS` (Inexact Solver).

In classic Newton method and proximal Newton method (Lee et al., 2012) it is common to consider inexact search for the next optimization direction. Since here we have chosen to use an iterative scheme for solving the subproblem (5), it seems natural to consider inexact solution for the inner optimization problem of NIM as well. However, in incremental optimization we do not have access to the full gradient of the function f, and hence we need to find a new stopping criterion for inexact model minimization in NIM. This criterion, from the one hand, must be eligible for calculation in the incremental setting and, from the other hand, must provide the rate of convergence similar to NIM with exact model minimization. Now we derive such criterion using notions from (Nesterov, 2013) and leave its theoretical analysis to section 3.

Problem (5) can be viewed as the minimization of a composite function: $\min_{y\in\mathbf{R}^d}[s(y) + h(y)]$, where $s(y) := \frac{1}{2}\|x - y\|_H^2$ is a smooth function whose gradient is Lipschitz-continuous with constant $\lambda_{\max}(H)$. Denote

$$T_L(x) := \operatorname*{argmin}_{y\in\mathbf{R}^d}\left[\nabla s(x)^\top y + \frac{L}{2}\|y-x\|^2 + h(y)\right]. \qquad (8)$$

Note that problem (8) can be solved exactly due to our initial assumptions on proximal mapping (2). Now let us consider the value

$$g_L(x) := L(x - T_L(x)),$$

which is an analogue of the gradient of a smooth function. If $h \equiv 0$, then $g_L(x)$ becomes $\nabla s(x)$ and does not depend on L. We propose the following stopping criterion for `IS`: if x satisfies

$$\|g_L(x)\| \le \min\{1, (\Delta_k)^\gamma\}\Delta_k,$$
$$\Delta_k := \|\bar{v}_k - \operatorname{prox}_h(\bar{v}_k - g_k)\|, \qquad (9)$$

then stop and return the point $\hat{x}_k := T_L(x)$ as an approximate minimizer of the model m_k. Here L is any number such that $L \ge L_0 \equiv 1$, $\bar{v}_k := \frac{1}{n}\sum_{i=1}^n v_k^i$, g_k is calculated using (6) and $\gamma \in (0, 1]$ is some parameter. Particular value of γ determines the local rate of convergence for inexact NIM method (see Section 3). Note that all the quantities in expression (9) are accessible for NIM since $T_L(x)$ can be calculated for any IS, g_k is calculated in an incremental way (7) and $\bar{v}_k$ can also be tracked in an incremental way (7).

Algorithm 1 NIM: a Newton-type Incremental Method

```
1: Input: $x_0, \dots, x_{n-1} \in \mathbf{R}^d$: initial points;
      $\alpha > 0$: step length.
2: Initialize model:
      $H \leftarrow (1/n) \sum_{i=1}^n \nabla^2 f_i(x_{i-1})$
      $u \leftarrow (1/n) \sum_{i=1}^n \nabla^2 f_i(x_{i-1}) x_{i-1}$
      $g \leftarrow (1/n) \sum_{i=1}^n \nabla f_i(x_{i-1})$
      $v_i \leftarrow x_{i-1},\ i = 1, \dots, n$
3: for $k \geq n-1$ do
4:    Minimize model using IS with stopping criterion (9): $\hat{x} \approx \mathrm{prox}_h^H[H^{-1}(u-g)]$
5:    Make a step: $x_{k+1} \leftarrow \alpha \hat{x} + (1-\alpha) x_k$
6:    Update model:
         $i \leftarrow (k+1) \bmod n + 1$
         $H \leftarrow H + (1/n)[\nabla^2 f_i(x_{k+1}) - \nabla^2 f_i(v_i)]$
         $u \leftarrow u + (1/n)[\nabla^2 f_i(x_{k+1}) x_{k+1} - \nabla^2 f_i(v_i) v_i]$
         $g \leftarrow g + (1/n)[\nabla f_i(x_{k+1}) - \nabla f_i(v_i)]$
         $v_i \leftarrow x_{k+1}$
7: end for
```

The general scheme of NIM in given in Algorithm 1.

3. Convergence analysis

3.1. Local rate of convergence

Here we briefly introduce main ideas which lead to the theorem on local convergence rate for both exact and inexact variants of NIM. Full set of lemmas and theorems and their proofs are given in supplementary material.

We assume x^* is a solution of (1) with positive definite Hessian:

$$\nabla^2 f(x^*) = \frac{1}{n} \sum_{i=1}^n \nabla^2 f_i(x^*) \succeq \mu_f I, \quad \mu_f > 0.$$

We analyze NIM with unit step size $\alpha_k \equiv 1$ and cyclic order of component selection. Besides, we suppose all functions f_i have Lipschitz-continuous gradients with constant L_f and Lipschitz-continuous Hessians with constant M_f.

At every iteration inexact NIM makes the step $x_{k+1} = \hat{x}_k = \operatorname{argmin} m_k(x) + e_k$, where e_k is some nonzero vector due to the stopping criterion (9). Lemma 3 from the theoretical analysis in (Nesterov, 2013) connects the current error $\|e_k\|$ with $\|g_L(x)\|$ from the last step of `IS`:

$$\|e_k\| \leq \lambda_{\min}^{-1}(H_k) \left(1 + \frac{L_f}{L}\right) \|g_L(x)\|.$$

Then the stopping criterion (9) guarantees

$$\|e_k\| \leq \lambda_{\min}^{-1}(H_k)(2+L_f)^{2+\gamma} \left(\frac{1}{n} \sum_{i=1}^n \|v_k^i - x^*\|^2\right)^{(1+\gamma)/2}.$$

Lemma 1 (main estimate). *Let $k \geq n-1$ be the number of the current iteration. Assume the last n points $x_k, \dots, x_{k-n+1}$ are close enough to x^*:*

$$\|x_{k-i} - x^*\| \leq \frac{\mu_f}{2M_f}, \qquad i = 0, \dots, n-1.$$

Then for the next generated point x_{k+1}:

$$\|x_{k+1} - x^*\| \leq \frac{M_f}{\mu_f} \left(\frac{1}{n} \sum_{i=0}^{n-1} \|x_{k-i} - x^*\|^2\right) + E \left(\frac{1}{n} \sum_{i=0}^{n-1} \|x_{k-i} - x^*\|^2\right)^{(1+\gamma)/2},$$

where $E = 0$ when the subproblem (5) is solved exactly and $E = \sqrt{\frac{8(2+L_f)^{5+2\gamma}}{\mu_f^3}}$ when it is solved using `IS` with stopping criterion (9).

This lemma shows that the local convergence rate of NIM is determined by the convergence rate of the following (recurrent) sequence:

$$z_k := A \left(\frac{1}{n} \sum_{i=1}^n z_{k-i}^2\right) + E \left(\frac{1}{n} \sum_{i=1}^n z_{k-i}^2\right)^{(1+\gamma)/2}, \ k \geq n,$$

where $A > 0$, $E \geq 0$, $0 < \gamma \leq 1$ are some constants.

For the sequence $\{z_k\}$ it is possible to show that, being properly initialized close enough to zero, it converges to zero at a Q-superlinear rate. Thus we come to the following main result on the local convergence rate of NIM:

Theorem 1 (local convergence rate). *Suppose that all the initial points $x_0, \dots, x_{n-1}$ are close enough to x^*:*

$$\|x_i - x^*\| \leq R, \qquad i = 0, \dots, n-1.$$

Then the sequence of iterates $\{x_k\}_{k \geq 0}$, generated by NIM with the unit step size and cyclic order of component selection, converges to x^* at an R-superlinear rate, i. e. there exists $\{z_k\}_{k \geq 0}$ such that

$$\|x_k - x^*\| \leq z_k, \qquad k \geq 0$$
$$z_{k+1} \leq q_k z_k, \qquad k \geq n,$$

where $q_k \to 0$ and

$$z_0 = \dots = z_{n-1} := \max_{0 \leq i \leq n-1} \|x_i - x^*\|.$$

If the subproblem (5) is solved exactly, then

$$R := \frac{\mu_f}{2M_f}, \qquad q_k := \left(1 - \frac{3}{4n}\right)^{2^{\lfloor k/n \rceil - 1}}.$$

Otherwise, if it is solved inexactly using `IS` *with termination condition* (9)*, then*

$$R := \min\left\{\frac{\mu_f}{2M_f}, \left(\frac{\mu_f^3}{128(2+L_f)^{5+2\gamma}}\right)^{1/(2\gamma)}\right\},$$

$$q_k := \left(1 - \frac{7}{16n}\right)^{(1+\gamma)^{\lfloor k/n \rceil}/2}.$$

3.2. Global convergence

Theorem 2 (global rate of convergence)**.** *Suppose that the gradients* ∇f_i*,* $i = 1, \dots, n$*, are Lipschitz-continuous. Let* $\{x_k\}_{k\geq 0}$ *be the sequence of iterates generated by NIM with a constant step length* $\alpha_k \equiv \alpha$*, where* α *is small enough, and cyclic order of component selection. Then for any initialization of* x_i*,* $i = 0, \dots, n-1$*, the sequence* $\{x_k\}$ *converges to* x^* *at a linear rate:*

$$\|x_k - x_*\|_2 \leq \text{const}\cdot c^k \, \|x_0 - x^*\|_2 , \qquad k = 0, 1, \dots,$$

where $c \in (0, 1)$.

The details are given in supplementary material.

4. Implementation details

4.1. Model initialization

One way to initialize the model is to set all the initial points equal: $x_1 = \dots = x_{n-1} = x_0$. However, this approach is quite ineffective because the method will not make any progress during the computation of the initial values for H, u and g (which takes a full pass over all the functions). A better strategy is to perform $n-1$ iterations of any other incremental method (such as SGD) from x_0 to obtain the points $x_1, \dots, x_{n-1}$ and build up the quantities H, u and g during the process. Another possibility is to start from $H = 0$, $u = 0$, $g = 0$ and then perform $n-1$ iterations of the main loop of NIM but without subtracting the quantities $\nabla^2 f_i(v_i)$, $\nabla^2 f_i(v_i)v_i$, $\nabla f_i(v_i)$ during the model updates. In our experiments we use the third strategy.

4.2. Order of component selection

We have experimented with two standard strategies of choosing the component i_k to update: 1) *cyclic* when $i_k = (k \bmod n) + 1$, and 2) *randomized* when at every iteration $i_k \in \{1, \dots, n\}$ is chosen uniformly at random. In all our experiments we observed that NIM always converges faster under the cyclic order. This result differs NIM from many other incremental methods where the randomized order usually works better (see also the discussion in Section 5.2 of (Defazio, 2014)).

It is possible to prove that in case of randomized order NIM convergence rate cannot be faster than linear. For this let us consider a particular function $\phi(x)$:

$$\phi(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) = \frac{1}{2}\|x\|^2 + \frac{1}{3}\|x\|^3,$$

$$f_1(x) = \frac{1}{2}\|x\|^2 + \frac{n}{3}\|x\|^3, \; f_i(x) = \frac{1}{2}\|x\|^2, i > 1.$$

Using NIM with randomized order for this function leads to the following lower bound:

$$\mathbf{E}[\|x_{k+1}\|] \geq \frac{1}{3}\left(1 - \frac{1}{n}\right)^k \mathbf{E}[\|v_0^1\|^2] \tag{10}$$

proving that NIM can have at most a linear convergence rate. Meanwhile using NIM with cyclic order for this function gives the following upper bound:

$$\|x_{k+1}\| \leq \|v_k^1\|^2, \tag{11}$$

that is equivalent to Q-quadratic convergence w.r.t epochs and R-superlinear convergence w.r.t. iterations. Details of derivations (10) and (11) are given in supplementary material. In case of random order some components of the model $m_k(x)$ may not be updated for a reasonable amount of time. This may lead to quite inaccurate models that spoil the convergence rate of NIM.

4.3. Linear models

In many machine learning problems the functions f_i have the following linearly-parameterized form: $f_i(x) := \phi_i(a_i^\top x)$, where $\phi_i : \mathbf{R} \to \mathbf{R}$ is a univariate function and $a_i \in \mathbf{R}^d$ is some known vector. In this case, by exploiting the structure of the functions f_i, we can substantially reduce the storage cost of NIM. Namely, denoting $\nu_k^i := a_i^\top v_k^i$, we can rewrite (7) as follows:

$$H_{k+1} = H_k + \frac{1}{n}\left[\phi_i''(\nu_{k+1}^i) - \phi_i''(\nu_k^i)\right] a_i a_i^\top,$$

$$u_{k+1} = u_k + \frac{1}{n}\left[\phi_i''(\nu_{k+1}^i)\nu_{k+1}^i - \phi_i''(\nu_k^i)\nu_k^i\right] a_i,$$

$$g_{k+1} = g_k + \frac{1}{n}\left[\phi_i'(\nu_{k+1}^i) - \phi_i'(\nu_k^i)\right] a_i.$$

Thus, instead of storing n vectors v_k^i, we need to store only n scalars ν_k^i. This reduces the memory requirements of NIM from $O(nd + d^2)$ to $O(n + d^2)$.

The stopping criterion (9) in IS uses $\bar{v}_k = \frac{1}{n}\sum_{i=1}^n v_k^i$. Calculating this value requires storage of all vectors v_k^i. In order to reduce the memory cost in this case we use x_k instead of $\bar{v}_k$ in the stopping criterion (9). In practice we found that this stopping criterion works well.

5. Experiments

We compare the empirical performance of NIM with that of several other methods for solving (1) on the binary logistic regression problem.

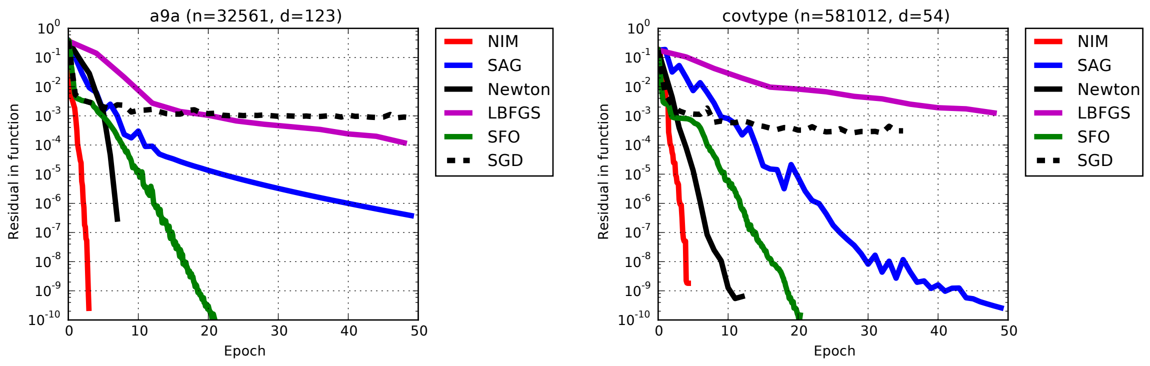

Figure 1. Empirical performance of different optimization methods w.r.t. epochs. Left: dataset *a9a*, right: dataset *covtype*.

We use the following methods in our comparison:

- *NIM*: the proposed method NIM (the version for linear models with the unit step size and cyclic order of component selection).
- *SGD*: a stochastic (proximal) gradient method (with step size $\alpha_k = n/(10k)$) as a representative of sublinearly convergent incremental methods.
- *SAG*: a proximal variant of the stochastic average gradient method of (Schmidt et al., 2013) (the version for linear models with constant step size $\alpha = 1/L$, where L is the analytically calculated Lipschitz constant) as a representative of linearly convergent incremental methods.
- *Newton*: an (inexact) proximal Newton method (Lee et al., 2012) (with the unit step size and true Hessian) as a non-incremental version of NIM.

When the objective is smooth, i.e. $h(x)$ is continuously differentiable, we also consider the following methods:

- *L-BFGS*: the limited-memory BFGS method (Liu & Nocedal, 1989) (with the history of size 10).
- *SFO*: the sum of functions optimizer (Sohl-Dickstein et al., 2014) (with default parameters).

All the methods except SFO are implemented in C++; for SFO we use its public implementation in Python[2].

As training data, we use 6 datasets (*a9a*, *covtype*, *protein*, *SUSY*, *mnist8m*) including 2 datasets with n of order of several millions. These datasets can be obtained from the LIBSVM website[3] and from Pascal Large Scale Learning Challenge 2008[4]. To binarize the original *mnist8m* dataset, we group digits with labels 0, 1, 2, 3, 4 into the first class, and other digits into the second class.

We set the regularization coefficient λ to $1/n$ and run all methods from $x_0 = 0$.

5.1. ℓ_2-regularized logistic regression

In this case our optimization problem is as follows:

$$\min_{x \in \mathbf{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ln(1 + \exp(a_i^\top x)) + \frac{\lambda}{2} \|x\|_2^2 . \qquad (12)$$

This problem corresponds to (1) with

$$f_i(x) := \ln(1 + \exp(a_i^\top x)), \quad h(x) := \frac{\lambda}{2} \|x\|_2^2 .$$

Since $h(x)$ is a quadratic function we use here for IS a conjugate gradient method.

Figure 1 shows empirical performance of different methods w.r.t. iterations (epochs) on two datasets. From this figure it is clear that superlinearly-convergent methods (NIM, Newton) outperform others by far in terms of number of iterations. We observe the similar picture on all other datasets, so due to the lack of space we do not include these results into the paper. Also it should be noted that for NIM only 5 epochs was enough for convergence with very high accuracy in all datasets in our experiments. Besides, the average number of IS iterations was in the interval $[1.25, 2]$. These observations give us a good preliminary estimation for the total amount of time needed for solving a particular problem using NIM.

We have experimented with different mini-batch sizes for NIM and SAG. Bigger mini-batches usually increase the

[2]SFO operates with (1) through big mini-batches, thus SFO python implementation may be fast enough due to fast vector operations in numpy

[3]`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[4]`http://largescale.ml.tu-berlin.de/about/`

Table 1. Calculation time of different methods for the problem (12). In the first column optimization accuracy is shown. Symbol „—“ means that the corresponding calculation time sufficiently exceeds the calculation time for NIM with accuracy 10^{-10}.

	a9a (n=32561, d=123)					*covtype (n=581012, d=54)*					*protein (n=145751, d=74)*				
Res	NIM	SAG	Newton	LBFGS	SFO	NIM	SAG	Newton	LBFGS	SFO	NIM	SAG	Newton	LBFGS	SFO
10^{-1}	**.01s**	.01s	.31s	.05s	.03s	.19s	.33s	.84s	.54s	**.04s**	**.02s**	.15s	.66s	.66s	.03s
10^{-2}	**.02s**	.05s	.56s	.10s	.08s	.51s	.96s	1.78s	1.77s	**.25s**	**.04s**	.33s	1.32s	.93s	.13s
10^{-4}	**.15s**	.19s	.81s	.43s	.98s	**.86s**	2.45s	3.09s	10.73s	3.80s	**.41s**	3.38s	2.16s	1.99s	.96s
10^{-6}	**.24s**	.66s	.93s	1.11s	1.57s	**1.49s**	4.12s	4.57s	31.84s	6.81s	**.59s**	14.26s	2.43s	4.52s	1.55s
10^{-8}	**.31s**	1.46s	1.04s	1.82s	2.18s	**1.92s**	5.90s	6.52s	—	9.86s	**.71s**	—	2.67s	5.94s	2.01s
10^{-10}	**.34s**	2.38s	1.04s	2.61s	2.81s	**2.12s**	9.97s	8.84s	—	12.44s	**.77s**	—	—	7.45s	2.49s

	SUSY (n=5000000, d=18)				*alpha (n=500000, d=500)*				*mnist8m (n=8100000, d=784)*			
Res	NIM	SAG	Newton	LBFGS	NIM	SAG	Newton	LBFGS	NIM	SAG	Newton	LBFGS
10^{-1}	**.09s**	2.71s	2.48s	1.78s	1.91s	**1.36s**	1.6m	4.01s	57.68s	**34.91s**	47.8m	1.1m
10^{-2}	**.13s**	3.84s	4.30s	2.52s	13.37s	**6.72s**	2.6m	17.68s	**1.6m**	2.1m	1.4h	5.2m
10^{-4}	**1.36s**	1.3m	11.33s	2.60s	36.65s	**36.04s**	3.4m	58.35s	16.7m	**7.1m**	—	1.6h
10^{-5}	**2.78s**	1.9m	14.43s	4.09s	**46.66s**	1.0m	3.6m	1.4m	**26.7m**	1.0h	—	—
10^{-6}	**3.95s**	2.2m	16.71s	5.26s	**53.92s**	1.5m	4.0m	1.9m	**33.5m**	—	—	—
10^{-8}	**5.30s**	2.6m	19.41s	8.43s	**1.0m**	2.7m	4.1m	2.8m	**46.0m**	—	—	—
10^{-10}	**5.95s**	3.4m	20.80s	9.01s	**1.2m**	4.3m	4.7m	3.4m	**53.3m**	—	—	—

number of iterations needed for convergence since the model $m_k(x)$ in NIM and similar model for SAG are updated less often w.r.t. iterations. However, bigger mini-batches may lead to faster convergence w.r.t. time since the scaled proximal mapping (5) is computed less often. For the problem (12) the optimal mini-batch size for SAG is 10, while for NIM the optimal mini-batch size is 100. With these mini-batch sizes the calculation time needed for model update and for the scaled proximal mapping (5) are comparable.

Table 1 shows the calculation time of different methods for the problem (12). Since SGD requires a huge number of iterations for convergence with high accuracy, it is not shown in the table. We also do not show SFO results for big datasets because its Python implementation is quite slow in this case.

Table 1 shows that NIM converges faster than other methods in almost all the cases. Here high iteration cost for NIM (at best $O(d^2)$) is compensated with very small number of iterations required for convergence. However, this is true only for datasets with relatively small d. Higher values of d lead to much higher iteration cost and prohibitive memory cost.

5.2. ℓ_1-regularized logistic regression

We consider the following optimization problem:

$$\min_{x\in\mathbf{R}^d} \quad \frac{1}{n}\sum_{i=1}^{n}\ln(1+\exp(a_i^\top x)) + \lambda\left\|x\right\|_1 . \qquad (13)$$

In this case

$$f_i(x) := \ln(1+\exp(a_i^\top x)), \quad h(x) := \lambda\left\|x\right\|_1 .$$

Since $h(x)$ is a non-differentiable function we use here FGM method for IS.

Methods L-BFGS and SFO do not support a proximal operator. Therefore we excluded them from experiments for the problem (13).

Concerning the number of iterations for the problem (13) empirical performance of different methods is quite similar to the previous case of l_2-regularizer. Again superlinearly-convergent methods (NIM, Newton) require much less iterations for convergence with high accuracy in comparison with other methods. However, here the average number of FGM iterations lies in rather big interval. Hence prediction for the total amount of time needed for NIM becomes a more challenging problem.

Also we have experimented with different mini-batch sizes for NIM and SAG. Here the scaled proximal mapping (5) requires more calculation time comparing to the case of l_2-regularizer. As a result, the optimal mini-batch size for NIM is 5000, while for SAG the optimal mini-batch size remains equal to 10.

Table 2 shows the calculation time of different methods for the problem (13). Here we see that for the problem (13) NIM as before converges faster than other methods in almost all the cases.

Table 2. Calculation time of different methods for the problem (13). In the first column optimization accuracy is shown. Symbol „—“ means that the corresponding calculation time sufficiently exceeds the calculation time for NIM with accuracy 10^{-10}.

	a9a			*covtype*			*protein*			*alpha*			*mnist8m*		
Res	NIM	SAG	Newton	NIM	SAG	Newton	NIM	SAG	Newton	NIM	SAG	Newton	NIM	SAG	Newton
10^{-1}	.12s	**.03s**	.46s	.48s	**.42s**	1.00s	**.04s**	.12s	.66s	26.76s	**1.31s**	1.1m	15.7m	**33.62s**	53.6m
10^{-3}	.38s	**.13s**	1.24s	**1.51s**	3.67s	2.87s	**.85s**	2.48s	2.07s	55.56s	**17.26s**	2.3m	46.9m	**4.0m**	2.5h
10^{-4}	.44s	**.22s**	1.40s	**2.20s**	4.49s	4.55s	**1.09s**	—	2.79s	1.1m	**35.51s**	2.5m	1.0h	**7.3m**	3.1h
10^{-5}	**.49s**	.64s	1.51s	**4.03s**	6.47s	6.53s	**1.22s**	—	3.07s	1.3m	**1.0m**	2.9m	**1.2h**	1.4h	—
10^{-6}	**.55s**	.95s	1.63s	**4.89s**	11.07s	8.24s	**1.36s**	—	3.49s	**1.3m**	1.5m	3.1m	**1.5h**	—	—
10^{-8}	**.63s**	2.26s	1.78s	**5.64s**	21.03s	9.95s	**1.59s**	—	3.96s	**1.5m**	2.9m	3.5m	**2.3h**	—	—
10^{-10}	**.69s**	4.09s	1.97s	**5.97s**	29.65s	10.86s	**1.74s**	—	4.42s	**1.6m**	4.8m	4.5m	**3.4h**	—	—

6. Discussion

In case $n = 1$ the proposed method NIM reduces to the proximal Newton method (Lee et al., 2012) and with additional condition $h(x) \equiv 0$ reduces to the classical Newton method. For the class of strongly convex functions with Lipschitz-continuous Hessians these methods are known to have local quadratic convergence and their inexact versions are known to have local superlinear convergence for proper choice of the forcing sequence η_k. Also these methods are globally convergent for small enough step size. Theorems 1 and 2 show that the same results are true for the proposed incremental version of Newton method. Moreover, convergence radius R for NIM is almost the same as that for classical Newton (Nesterov, 2004), in particular, it does not depend on number of functions n. These facts support NIM to be a proper extension of Newton-like methods in incremental setting.

From global convergence perspective, theorem 2 suggests taking a step size proportional to $(\mu_f/L_f)^3$. Unfortunately, for practical purposes this value is usually too small. However, this result should be considered mostly as a qualitative one (the method is globally convergent with a linear rate). In practice, NIM successfully converges with much larger step sizes (for example, unity step size is uniformly appropriate for logistic regression). The reasons for such a small step size in the theorem are the following. The standard global convergence analysis of the classical Newton method (in terms of μ and L) tells us that the step size should be of order μ/L in comparison with $1/L$ for the classical gradient method (see e.g. (Boyd & Vandenberghe, 2004)). The problem here is that the analysis does not use the fact that the matrix in the method is the Hessian; it only considers some matrix with eigenvalues bounded between μ and L. As soon as the true Hessian is considered (like in cubic regularization from (Nesterov & Polyak, 2006)) Newton method becomes more effective in comparison with gradient method even in the first stage of optimization (far from solution). Our analysis of NIMs global convergence is based on IAG analysis from (Gurbuzbalaban et al., 2015), and we use only some scaling matrix with bounded eigenvalues instead of a true average estimate for the Hessian. As a result, we get a step size of several orders less than one for IAG, which in turn is less than step size for SAG. Hence, a small step size for NIM global convergence is mostly an issue of the current theoretical analysis than a reflection of a true NIM behavior.

Although in all our experiments with logistic regression NIM with unit step length always converged starting from $x_0 = 0$, in other settings a step size adaptation were vital for convergence. However, in incremental situation a step size adaptation procedure is a challenging problem, since we do not have access to the true objective ϕ and thus cannot use the Armijo condition or trust-region approach. The possible way-out here is to consider a cubic regularization in model $m_k(x)$ similar to (Nesterov & Polyak, 2006). In this case the model $m_k(x)$ becomes a strict upper bound for the objective ϕ, and hence the algorithm with unit step size becomes globally-convergent. Moreover, cubic regularization gives possibility to solve problems (1) with non-convex objective. We refer this as one of directions for future research.

7. Acknowledgments

This work was supported by RFBR project No. 15-31-20596 (mol-a-ved) and by Microsoft: Moscow State University Joint Research Center (RPD 1053945).

References

Andrew, G. and Gao, J. Scalable training of L1-regularized log-linear models. In Ghahramani, Zoubin (ed.), *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pp. 33–40. Omnipress, 2007.

Bertsekas, Dimitri P. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38, 2011.

Blatt, Doron, Hero, Alfred O, and Gauchman, Hillel. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

Bordes, Antoine, Bottou, Léon, and Gallinari, Patrick. SGD-QN: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009.

Bousquet, Olivier and Bottou, Léon. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pp. 161–168, 2008.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge Univesity Press, 2004.

Byrd, Richard H, Hansen, SL, Nocedal, Jorge, and Singer, Yoram. A stochastic quasi-newton method for large-scale optimization. *arXiv preprint arXiv:1401.7020*, 2014.

Defazio, Aaron. *New Optimisation Methods for Machine Learning*. PhD thesis, Australian National University, 2014. http://www.aarondefazio.com/pubs.html.

Defazio, Aaron, Bach, Francis, and Lacoste-Julien, Simon. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014a.

Defazio, Aaron J, Caetano, Tibério S, and Domke, Justin. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of 31st International Conference on Machine Learning*, 2014b.

Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software, Vol. 33(1)*, 2010. URL http://www.stanford.edu/~hastie/Papers/glmnet.pdf.

Gurbuzbalaban, M., Ozdaglar, A., and Parrilo, P. On the convergence rate of incremental aggregated gradient algorithms. *arXiv preprint arXiv:1506.02081*, 2015.

Gürbüzbalaban, Mert, Ozdaglar, Asuman, and Parrilo, Pablo. A globally convergent incremental newton method. *Mathematical Programming*, 151(1):283–313, 2015.

Johnson, Rie and Zhang, Tong. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Kolte, R., Erdoglu, M., and Oezguer, A. Accelerating svrg via second-order information. *NIPS Workshop on Optimization for Machine Learning*, 2015.

Lee, Jason D, Sun, Yuekai, and Saunders, Michael. Proximal newton-type methods for convex optimization. In Pereira, F., Burges, C.J.C., Bottou, L., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 827–835. Curran Associates, Inc., 2012.

Liu, Dong C and Nocedal, Jorge. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Lucchi, A., McWilliams, B., and Hofmann, T. A variance reduced stochastic newton method. *arXiv preprint arXiv:1503.08316*, 2015.

Mairal, Julien. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

Moritz, F., Nishihara, R., and Jordan, M. A linearly-convergent stochastic l-bfgs algorithm. *arXiv preprint arXiv:1508.02087*, 2015.

Nesterov, Y. *Introductory Lectures on Convex Optimization*. Springer, 2004.

Nesterov, Y. Gradient methods for minimizing composite functions. *Mathematical Programming, V.140(1)*, pp. 125–161, 2013.

Nesterov, Y. and Polyak, B.T. Cubic regularization of newton method and its global performance. *Mathematical Programming, V.108(1)*, pp. 177–205, 2006.

Schmidt, Mark, Roux, Nicolas Le, and Bach, Francis. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

Schraudolph, Nicol N, Yu, Jin, and Günter, Simon. A stochastic quasi-newton method for online convex optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 436–443, 2007.

Sohl-Dickstein, Jascha, Poole, Ben, and Ganguli, Surya. Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 604–612, 2014. URL http://jmlr.org/proceedings/papers/v32/sohl-dicksteinb14.html.