QuickeNing: A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization

Hongzhou Lin, Julien Mairal, Zaid Harchaoui

QuickeNing: A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization*

Hongzhou Lin
Inria
hongzhou.lin@inria.fr

Julien Mairal
Inria
julien.mairal@inria.fr

Zaid Harchaoui
University of Washington
zaid@uw.edu

October 4, 2016

Abstract

We propose an approach to accelerate gradient-based optimization algorithms by giving them the ability to exploit curvature information using quasi-Newton update rules. The proposed scheme, called QuickeNing, is generic and can be applied to a large class of first-order methods such as incremental and block-coordinate algorithms; it is also compatible with composite objectives, meaning that it has the ability to provide exactly sparse solutions when the objective involves a sparsity-inducing regularization. QuickeNing relies on limited-memory BFGS rules, making it appropriate for solving high-dimensional optimization problems; with no line-search, it is also simple to use and to implement. Besides, it enjoys a worst-case linear convergence rate for strongly convex problems. We present experimental results where QuickeNing gives significant improvements over competing methods for solving large-scale high-dimensional machine learning problems.

1 Introduction

Convex composite optimization arises in many scientific fields, such as image and signal processing or machine learning. It consists of minimizing a real-valued function composed of two convex terms:

$$\min_{x\in\mathbb{R}^d} \Big\{ f(x) \triangleq f_0(x) + \psi(x) \Big\}, \tag{1}$$

where f_0 is smooth with Lipschitz continuous derivatives, and ψ is a regularization function which is not necessarily differentiable. For instance, the ℓ_1-norm $\psi(x) = \|x\|_1$ is very popular in image processing [13, 27] for encouraging sparse solutions; composite minimization also encompasses constrained minimization when considering extended-valued indicator functions ψ that may take the value $+\infty$ outside of a convex set $\mathcal{C}$ and 0 inside (see [20]). In general, algorithms that are dedicated to composite optimization only require to be able to compute efficiently the proximal operator of ψ:

$$p_\psi(y) \triangleq \operatorname*{arg\,min}_{x\in\mathbb{R}^d} \left\{ \psi(x) + \frac{1}{2}\|x-y\|^2 \right\},$$

where $\|\cdot\|$ denotes the Euclidean norm. Note that when ψ is an indicator function, the proximal operator corresponds to the simple Euclidean projection.

To solve (1), significant efforts have been devoted to (i) extending techniques for smooth optimization to deal with composite terms; (ii) exploiting the underlying structure of the problem—is f a finite sum of independent terms? Is ψ separable in different blocks of coordinates? (iii) exploiting the local curvature of

*This work was supported by ANR (MACARON project ANR-14-CE23-0003-01), the program "Learning in Machines and Brains" (CIFAR), the project Titan (CNRS-Mastodons), and the MSR-Inria joint centre.

the smooth term f to achieve faster convergence than gradient-based approaches when the dimension d is very large. Typically, the first point is well understood in the context of optimal first-order methods, see [1, 33], and the last point is tackled with effective heuristics such as L-BFGS when the problem is smooth [25, 34]. Yet, solving all these problems at the same time is challenging: this is precisely the focus of this paper.

In particular, a problem of interest that has first motivated our work is that of empirical risk minimization (ERM); the problem arises in machine learning and can be formulated as the minimization of a composite function $f : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\}, \tag{2}$$

where the functions f_i are convex and smooth with Lipschitz continuous derivatives, and ψ is a composite term, possibly non-smooth. The function f_i measures the fit of some model parameters x to a specific data point indexed by i, and ψ is a regularization penalty to prevent over-fitting. To exploit the sum structure of f, a large number of randomized incremental gradient-based techniques have been proposed, such as SAG [42], SAGA [9], SDCA [43], SVRG [44], Finito [10], or MISO [26]. These approaches access a single gradient $\nabla f_i(x)$ at every iteration instead of the full gradient $(1/n) \sum_{i=1}^{n} \nabla f_i(x)$ and achieve lower computational complexity in expectation than optimal first-order methods [1, 33] under a few assumptions. Yet, these methods are unable to exploit the curvature of the objective function; this is indeed also the case for variants that are accelerated in the sense of Nesterov [15, 24, 43].

To tackle (2), dedicated first-order methods are often the default choice in machine learning, but it is also known that generic Quasi-Newton approaches are sometimes surprisingly effective in the smooth case—that is when $\psi = 0$, see, e.g., [42] for extensive benchmarks. Since the dimension of the problem d is typically very large ($d \geq 10\,000$), "limited memory" variants of these algorithms, such as L-BFGS, are necessary to achieve the desired scalability [25, 34]. The theoretical guarantees offered by L-BFGS are somewhat weak, meaning that it does not outperform accelerated first-order methods in terms of worst-case convergence rate, and also it is not guaranteed to correctly approximate the Hessian of the objective. Yet, it remains one of the greatest practical success of smooth optimization, and adapting it efficiently to composite and structured problems such as (2) is of utmost importance nowadays.

For instance, there have been several attempts to develop a proximal Quasi-Newton method [6, 22, 40, 45], but they typically require computing many times the proximal operator of ψ with respect to a variable metric, which may be as computationally demanding as solving the original problem. More related to our work, L-BFGS is combined with SVRG for minimizing smooth finite sums in [18]. The goal of our paper is more general since it is not limited to SVRG, but it can be applied to a large-class of first-order techniques for composite optimization, including other incremental algorithms [9, 10, 26, 42, 43] and block coordinate descent methods [36, 37].

More precisely, our main contribution is a generic meta-algorithm, called QuickeNing (the letters "Q" and "N" stand for Quasi-Newton), which uses a given optimization method to solve a sequence of auxiliary problems up to some specific accuracy, resulting in faster global convergence in practice. The resulting scheme admits a simple interpretation: *it may be seen as applying an L-BFGS algorithm with inexact (but accurate enough) gradients to the Moreau-Yosida regularization of the objective.* As a result, our approach is (i) generic, as stated previously; (ii) despite the smoothing of the objective, the sub-problems that we solve are composite ones, which may lead to exactly sparse iterates when a sparsity-inducing regularization is involved, e.g., the ℓ_1-norm; (iii) it admits a worst-case linear convergence rate for strongly convex problems, which is typically the best guarantees obtained for L-BFGS schemes in the literature; (iv) it is easy to use and does not require using a line search algorithm, which is sometimes computationally expensive and difficult to calibrate in classical Quasi-Newton methods.

The idea of combining second-order or quasi-Newton methods with Moreau-Yosida regularization is in fact relatively old. It may be traced back to variable metric proximal bundle methods [8, 17, 28], which use BFGS updates on the Moreau-Yosida smoothing of the objective and bundle methods to approximately solve the corresponding sub-problems. Our approach revisits this principle with *a limited-memory variant* (to deal with large dimension d), with *an alternative strategy to line search schemes*—which is useful when f

is a large sum of n functions as in (2)—and with *a global complexity analysis* that is more relevant than convergence rates that do not take into account the cost per iteration.

To demonstrate the effectiveness of our scheme in practice, we evaluate our algorithm on the logistic regression and Elastic-net formulations [46], which include both smooth and non smooth objective functions. We use large-scale machine learning datasets and show that our algorithm performs at least as well as Catalyst and the classical L-BFGS scheme in all experiments, and significantly outperforms them in many cases, especially in ill-conditioned ones.

The paper is organized as follows: Section 2 presents related work on Quasi-Newton methods such as L-BFGS; we introduce our algorithm in Section 3 as well as basic properties of the Moreau-Yosida regularization, and we provide a convergence analysis in Section 4; Section 5 is devoted to experiments and Section 6 concludes the paper.

2 Related work

The history of quasi-Newton methods can be traced back to the 1950's [3, 21, 35]. Quasi-Newton methods often lead to significantly faster convergence in practice compared to simpler gradient-based methods for solving smooth optimization problems [41]. Yet, a theoretical analysis of quasi-Newton methods that explains their impressive empirical behavior on a wide range of problems is still an open topic. Here, we review briefly the well-known BFGS algorithm in Section 2.1, its limited memory variant [34], and a few recent extensions. Then, we present earlier works that combine proximal point and Quasi-Newton algorithms in Section 2.2, which is related to the strategy we use in our paper.

2.1 Quasi-Newton methods for smooth optimization

The most popular Quasi-Newton method is BFGS, named after its inventors (Broyden-Fletcher-Goldfarb-Shanno), and its limited variant L-BFGS [35]. These approaches will be the workhorses of the QuickeNing meta-algorithm. Consider a smooth convex objective f to be minimized, the BFGS method successively constructs a couple (x_k, B_k) with the following update:

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k) \quad \text{and} \quad B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}, \tag{3}$$

where α_k is a suitable stepsize and

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k).$$

The condition $y_k^T s_k > 0$ is satisfied as soon as f is strongly convex. To determine the stepsize α_k, Wolfe's line-search is a simple choice which provides a linear convergence rate in the worst case. If, in addition, the objective is twice differentiable and the Hessian is Lipschitz continuous, the convergence is asymptotically superlinear [35].

The limited memory variant L-BFGS [34] overcomes the issue of storing B_k for large d, by replacing it by another positive definite matrix—say $\tilde{B}_k$—which can be build from a "generating list" of at most l pairs of vectors $\{(s_i^k, y_i^k)\}_{i=1\dots j}$ along with an initial diagonal matrix $\tilde{B}_1$. Formally, $\tilde{B}_k$ can be computed by applying at most l times a recursion similar to (3) involving pairs of the generating list. Between iteration k and $k+1$, the generating list is incrementally updated, by removing the oldest pair in the list and adding a new one. What makes the approach appealing is the ability of computing $B_k^{-1} z$ for any vector z with only $O(ld)$ floating point operations instead of $O(d^3)$ for a naive implementation with matrix inversion. The price to pay is that superlinear convergence becomes out of reach in contrast to BFGS.

L-BFGS is thus appropriate for high-dimensional problems (when d is large), but it still requires computing the full gradient at each iteration, which may be cumbersome in the large sum setting (2). This motivated stochastic counterparts of the Quasi-Newton method (SQN) [5]. The direct application substituting the full gradient $\nabla f(x_k)$ by a stochastic estimate is unfortunately not convergent. Instead, the SQN method [5] uses

a product of a sub-sampled Hessian and s_k to approximate y_k. SQN can be complemented by a variance reduction scheme such as SVRG [18, 29].

2.2 Combining the proximal point algorithm and Quasi-Newton

There are several ways to extend Quasi-Newton methods for nonsmooth optimization. A first approach consists in minimizing successive quadratic approximations. A major drawback is then the computation of the proximal operator. Since B_k is dense and changes over the iterations, an exact solution to the proximal operator is usually not available. An inexact variant has been analyzed in [6], but it is unclear whether the quadratic approximation could be solved inexactly in an efficient manner.

A second approach consists in applying Quasi-Newton methods after Moreau-Yosida smoothing or equivalently a proximal point algorithm on the objective function [4, 8, 16, 17]. The underlying idea is to first smooth and improve the conditioning of the objective function, then apply a Quasi-Newton method on the resulting surrogate objective function. This implies minimizing a sequence of sub-problems. For instance, in [8], the sub-problems are solved inexactly by proximal bundle methods, and in [16] by using a sophisticated sufficient descent criterion.

The proposed QuickeNing algorithm is based on this idea—that is, first smoothing the objective using the Moreau-Yosida infimal-convolution then applying an appropriate first-order method to approximately minimize the resulting surrogate objective. In contrast to previous works, we give a convergence analysis taking into account the inner-loop complexity. Furthermore, we propose an effective strategy to remove the need of a line-search scheme and simple rules to set the parameters. Before going into details, we recall some remarkable properties of Moreau-Yosida regularization.

3 The QuickeNing meta-algorithm for composite optimization

In this section, we present the QuickeNing algorithm, which relies on fruitful blend of Moreau-Yosida regularization and quasi-Newton. We start by recalling well-known properties of the Moreau-Yosida regularization in Section 3.1 and giving a fresh look on the Catalyst algorithm of Lin et al. [24].

3.1 Basic properties of the Moreau-Yosida regularization

Let us consider the general optimization problem

$$\min_{x \in \mathbb{R}^p} f(x),$$

where f is assumed to be convex. The Moreau-Yosida regularization of the objective is defined as the minimum value of the proximal mapping

$$F(x) = \min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2} \|z - x\|^2 \right\}, \tag{4}$$

where κ is a positive scalar. The image $p(x)$ of x under the proximal mapping is defined as the solution of the previous problem, which is unique by strong convexity. The Moreau-Yosida regularization admits classical properties, which we present below; see [23] for more details and elementary proofs.

Proposition 1 (Basic properties of the Moreau-Yosida regularization)**.** *Let us consider the Moreau-Yosida regularization F of the convex function f defined in (4); Then, the following statements hold*

1. *F is convex and minimizing f and F are equivalent in the sense that*

$$\min_{x \in \mathbb{R}^p} F(x) = \min_{x \in \mathbb{R}^p} f(x),$$

and the solution set of the two above problems coincide with each other.

2. *F is continuously differentiable even when f is not and*

$$\nabla F(x) = \kappa(x - p(x)), \tag{5}$$

where $p(x)$ is the proximal mapping, which we have defined previously as the solution of (4). Moreover the gradient ∇F is Lipschitz continuous with constant $L_F = \kappa$.

3. *When f is μ-strongly convex with respect to the Euclidean norm, F is μ_F-strongly convex with constant $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$.*

4. *For all x in $\mathbb{R}^p$,*

$$F(x) \leq f(x). \tag{6}$$

Interestingly, F inherits all the convex properties of f and more importantly it is always continuously differentiable. These observations yield a simple strategy for minimizing any convex function f, by simply minimizing F with an algorithm that is able to handle *smooth* functions. Such an approach is appealing but it raises several difficulties. In particular, computing the gradient of F requires the exact solution $p(x)$ of (4), for which no closed-form is available in general. It is thus necessary to use an approximate solution, which implies defining an inexactness criterion that is easy to check and to control the accuracy of the gradient approximation to ensure convergence; see, e.g. [11].

3.2 A fresh look on the Catalyst algorithm

We recall a well-known equivalence between the Moreau-Yosida regularization and the proximal point algorithm.

3.2.1 The proximal point algorithm

Consider the classical gradient descent algorithm with step size $1/L_F = 1/\kappa$ to minimize F:

$$x_{k+1} = x_k - \frac{1}{\kappa}\nabla F(x_k).$$

By rewriting the gradient $\nabla F(x_k)$ as $\kappa(x_k - p(x_k))$, we obtain

$$x_{k+1} = p(x_k) = \operatorname*{arg\,min}_{z\in\mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|z - x_k\|^2 \right\}. \tag{7}$$

This is exactly the proximal point algorithm [38].

3.2.2 Catalyst and the accelerated inexact proximal point algorithm

Since gradient descent on F yields the proximal point algorithm, it is also natural to apply an accelerated first-order method to get faster convergence. To that effect, Nesterov's algorithm [30] uses a two-stage update:

$$x_{k+1} = y_k - \frac{1}{\kappa}\nabla F(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k),$$

where β_{k+1} is a specific extrapolation parameter (see [31]). We may now rewrite the update using the value of ∇F given in (5), which gives:

$$x_{k+1} = p(y_k) \quad \text{and} \quad y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k)$$

This is known as the accelerated proximal point algorithm introduced by Güler [19], which has been studied in the context of approximate solutions $p(y_k)$ [11, 19, 39]. The Catalyst algorithm of Lin et al. [24] is an extension of the previous approach with a practical inexactness criterion. The Catalyst inexactness criterion allows to control the complexity of computing the approximate gradient, that is solving the auxiliary problems (7). Besides, the criterion may be easily checked by computing duality gaps. The criterion is actually a key component of QuickeNing based on L-BFGS.

3.3 Combining L-BFGS and inexact gradients

A first idea consists in applying an accelerated first-order algorithm to the Moreau-Yosida smoothing F; this corresponds to the Catalyst algorithm [24]. We propose instead to perform L-BFGS updates using inexact gradients [14]. The procedure for computing $\nabla F(x)$ and the inexactness criterion are presented in Algorithm 1, which returns an approximation of the function value $F(x)$, and of the proximal mapping $z \approx p(x)$. When $\varepsilon = 0$, the gradient is exact. We remark that the criterion $h(z) - h^* \leq \varepsilon$ is weak, compared to other criteria in previous works [11, 19, 39]. The proposed criterion has a major advantage. The condition $h(z) - h^* \leq \varepsilon$ can easily be checked by computing duality gaps in practice. Furthermore, gradient-based methods often admit convergence rates that allow us to control the computational complexity for solving the sub-problems (8) up to accuracy ε.

Algorithm 1 Procedure GradientEstimate

input Current point x in $\mathbb{R}^p$; accuracy ε; smoothing parameter $\kappa > 0$.

1: Compute the approximate proximal mapping using an optimization method $\mathcal{M}$:

$$z \approx \operatorname*{arg\,min}_{v \in \mathbb{R}^d} \left\{ h(v) \triangleq f(v) + \frac{\kappa}{2}\|v - x\|^2 \right\}, \tag{8}$$

such that $h(z) - h^* \leq \varepsilon$ where $h^* = \min_{z \in \mathbb{R}^d} h(z)$; define $F_a = h(z)$.

2: Estimate the gradient of the Moreau-Yosida objective function

$$g = \kappa(x - z).$$

output approximate gradient estimate g, objective value F_a, proximal mapping z.

With a way to approximate the gradient of F in hand, we now present an appropriate quasi-Newton-type rule to update the metric along the iterations. In order to address high-dimensional problems, we propose to modify the classical L-BFGS update rule, as outlined in Algorithm 2. As shown later in Section 2.1, the proposed update rule, denoted L-BFGS for simplicity of the exposition, consists of updating incrementally a generating list of vectors $\{(s_i, y_i)\}_{i=1\dots j}$, which implicitly define the L-BFGS matrix H. We use here the two-loop recursion detailed in [35, Algorithm 7.4] We present the full QuickeNing scheme in Algorithm 3. Next, we make a few remarks regarding various properties of our approach.

Algorithm 2 Quasi-Newton-type update rule L-BFGS

input current L-BFGS matrix H formed from a generating list $\{(s_i, y_i)\}_{i=1\dots j}$ and initial diagonal matrix H_1; new candidate pair (s, y); L-BFGS parameters $0 < c_1, c_2 \leq 1$; memory parameter l;

1: **if** the following condition is satisfied

$$c_1 \mu_F \|s\|^2 < y^T s \quad \text{and} \quad \frac{c_2}{L_F}\|y\|^2 < y^T s. \tag{9}$$

then

2: add (s, y) to the generating list, and remove the oldest pair if the cardinal exceeds l.

3: **else**

4: keep the generating list unchanged.

5: **end if**

output new matrix H (generating list and H_0).

Handling composite objective functions In machine learning or signal processing, convex composite objectives (1) with a non-smooth penalty ψ are typically formulated to encourage solutions with specific

Algorithm 3 QuickeNing

input Initial point x_0 in $\mathbb{R}^p$; decreasing sequence $(\varepsilon_k)_{k\geq 0}$; number of iterations K; smoothing parameter $\kappa > 0$; L-BFGS parameters $0 < c_1, c_2 \leq 1$; optimization method $\mathcal{M}$.

Initialization: $(g_0, F_0, z_0) = \mathsf{GradientEstimate}(x_0, \varepsilon_0)$; BFGS matrix $H_0 = (1/\kappa)I$.

for $k = 0, \ldots, K-1$ **do**

Perform the Quasi-Newton step

$$x_{\text{test}} = x_k - H_k g_k.$$

Estimate the new gradient and the Moreau-Yosida function value

$$(g_{\text{test}}, F_{\text{test}}, z_{\text{test}}) = \mathsf{GradientEstimate}(x_{\text{test}}, \varepsilon_{k+1}).$$

if sufficient decrease is obtained

$$F_{\text{test}} \leq F_k - \frac{1}{4\kappa}\|g_k\|^2 + \varepsilon_k, \tag{10}$$

then

accept the new iterate: $(x_{k+1}, g_{k+1}, F_{k+1}, z_{k+1}) = (x_{\text{test}}, g_{\text{test}}, F_{\text{test}}, z_{\text{test}})$.

else

update the current iterate with the proximal mapping: $x_{k+1} = z_k$.

$$(g_{k+1}, F_{k+1}, z_{k+1}) = \mathsf{GradientEstimate}(x_{k+1}, \varepsilon_{k+1}).$$

end if

update $H_{k+1} = \mathsf{L\text{-}BFGS}(H_k, x_{k+1} - x_k, g_{k+1} - g_k)$.

end for

output last proximal mapping z_K (solution).

characteristics; in particular, the ℓ_1-norm is known to provide sparsity. Smoothing techniques [32] may allow us to solve the optimization problem up to some chosen accuracy, but they provide solutions that do not inherit the properties induced by the non-smoothness of the objective. To illustrate what we mean by this sentence, we may consider smoothing the ℓ_1-norm, leading to a solution vector with small coefficients, but not with exact zeroes. When the goal is to perform model selection—that is, understanding which variables are important to explain a phenomenon, exact sparsity is seen as an asset, and optimization techniques dedicated to composite problems such as FISTA [1] are often preferred.

Then, one may argue that our scheme operates on the smoothed objective F, leading to iterates $(x_k)_{k\geq 0}$ that suffer from the above "non-sparse" issue, assuming that ψ is the ℓ_1-norm. Yet, our approach also provides iterates $(z_k)_{k\geq 0}$ that are computed using the original optimization method $\mathcal{M}$ that we wish to accelerate. When $\mathcal{M}$ handles composite problems without smoothing, typically when $\mathcal{M}$ is a proximal block-coordinate, or incremental method, the iterates $(z_k)_{k\geq 0}$ may be sparse. For this reason, our theoretical analysis presented in Section 4 studies the convergence of the sequence $(f(z_k))_{k\geq 0}$ to the solution f^*.

On the absence of line-search scheme Another key property of the QuickeNing algorithm is that it does not require using a line-search scheme to select a step-size, which is typically necessary to ensure the monotonic descent (and convergence) of classical BFGS and L-BFGS algorithms [34]. In the context of the Moreau-Yosida regularization, any line-search would be prohibitive, since it would require to evaluate the function F multiple times, hence solving the sub-problems (8) as many times at every iteration.[1] Here, we choose a simple strategy that selects $x_{k+1} = z_k$ when the sufficient descent condition (10) is not satisfied. z_k is indeed a good candidate since it is obtained by performing one step of the inexact proximal point algorithm, which we have described above, and whose convergence properties are well understood. In practice, we have observed that the sufficient decrease condition is almost all the time satisfied, given the choice of parameters κ and ε_k provided in the experimental section.

Discussion about the modified L-BFGS rule The inexactness of the gradient computation requires changing the classical L-BFGS rule in order to guarantee the positive definiteness of the matrices H_k. This is typically achieved by skipping some L-BFGS updates that would make the resulting matrix non-positive definite [14]. In practice, the classical condition to ensure positive definiteness is to have $y^\top s > 0$ in Algorithm 2, see Section 2.1. Our convergence analysis presented in Section 4 suggests instead the stronger one (9), which represents strong-convexity and Lipschitz gradient inequalities when using exact gradients with $c_1 = c_2 = 1$. Intuitively, they are still valid when the gradient accuracy is good enough or when c_1, c_2 are small enough.

4 Convergence analysis

In this section, we study the convergence of the QuickeNing algorithm—that is, the rate of convergence of the quantities $f(z_k) - f^*$ and $F(x_k) - F^*$, and also its computational complexity, which takes into account the cost of solving the sub-problems (8). We start by stating our main convergence result for strongly convex objectives. The requirement about $\mathcal{M}$ is similar to that of Catalyst [24].

Proposition 2 (Complexity analysis for μ-strongly convex objectives)**.** *Assume that f is μ-strongly convex and that $\mathcal{M}$ is always able to produce a sequence of iterates $(w_t)_{t\geq 0}$ for solving the sub-problems (8) such that*

$$h(w_t) - h^* \leq A(1 - \tau_{\mathcal{M}})^t (h(w_0) - h^*) \quad \text{for some constants} \quad A, \tau_{\mathcal{M}} > 0. \tag{11}$$

Then, choose a sequence $\varepsilon_k = C(1-\rho)^{k+1}/3$ with $C \geq (f(x_0) - f^*)$ and define $\rho = \frac{\mu}{8(\mu+\kappa)}$; the sequence of

[1] Note that in the context of BFGS applied to F, a heuristic line-search that requires evaluating f instead of F is also proposed in [8], but this heuristic does not guarantee the algorithm convergence and is still computationally demanding when f is a large sum of functions as in (2).

iterates $(z_k)_{k\geq 0}$ and $(x_k)_{k\geq 0}$ produced by Algorithm 3 satisfy

$$\max\{F(x_k) - F^*, f(z_k) - f^*\} \leq \frac{C}{\rho}(1-\rho)^{k+2}. \tag{12}$$

Moreover, by initializing $\mathcal{M}$ with $w_0 = z_k$ at iteration k of Algorithm 3, each sub-problem (8) is solved up to accuracy ε_{k+1} in at most a constant number $T_{\mathcal{M}}$ of iterations of $\mathcal{M}$, where $T_{\mathcal{M}} = \tilde{O}(1/\tau_{\mathcal{M}})$.[2]

The linear convergence requirement for $\mathcal{M}$ is relatively standard and applies to many first-order methods. As we shall see, the proof follows in part that of Catalyst [24], but requires significant modifications to accommodate the L-BFGS metric, which will be presented later in this section. Importantly, as Catalyst, the proposition may also hold for randomized optimization methods $\mathcal{M}$ that provide a convergence rate in expectation, and also when the convergence rate is stated in terms of dual certificate instead of primal quantities $h(w_t) - h^*$. The next proposition provides such a result for the methods SDCA [43] and MISO/Finito [10, 24].

Proposition 3 (Complexity analysis for SDCA and MISO/Finito)**.** *When applying the randomized dual methods SDCA [43] or MISO/Finito [10, 24] to solve the sub-problems (8), they generate a sequence of iterates $(w_t)_{t\geq 0}$ such that*

$$\mathbb{E}[h(w_t) - d(w_t)] \leq A(1-\tau_{\mathcal{M}})^t(h(w_0) - d(w_0)) \quad \text{for some constants} \quad A, \tau_{\mathcal{M}} > 0. \tag{13}$$

where the function d is such that $d(w_t) \leq h^*$ for all t. Then, with the same assumptions as in Proposition 2, when initializing $w_0 = z_k + \frac{\kappa}{\kappa+\mu}(x_{k+1} - x_k)$ at iteration k of Algorithm 3, each sub-problem (8) is solved up to accuracy ε_{k+1} in $T_{\mathcal{M}}$ iterations of the method $\mathcal{M}$, with $\mathbb{E}[T_{\mathcal{M}}] = \tilde{O}(1/\tau_{\mathcal{M}})$.

What the two previous propositions tell us is that linearly-convergent methods $\mathcal{M}$, both in terms of primal objective function value or dual certificate, will also enjoy a linear convergence rate when used within the QuickeNing scheme. Specifically, to obtain an ε-accurate solution, the number of calls of $\mathcal{M}$ (or expected number of calls if $\mathcal{M}$ is non-deterministic) is at most

$$\tilde{O}\left(\frac{T_{\mathcal{M}}}{\rho}\log\left(\frac{1}{\varepsilon}\right)\right) = \tilde{O}\left(\frac{1}{\tau_{\mathcal{M}}\rho}\log\left(\frac{1}{\varepsilon}\right)\right).$$

We remark however that like classical L-BFGS algorithms, the theoretical complexity is a worst case and does not outperform other first-order methods [24]. In particular, a clever choice of κ in Catalyst achieves lower theoretical computational complexity than the original method $\mathcal{M}$—thus, resulting in theoretical *acceleration*. Here, the analysis suffers from a mismatch between theory and practice (like any L-BFGS method) and there is no choice of κ leading to theoretical acceleration. As discussed in the experimental section, choosing κ as in Catalyst [24] results nevertheless in significantly faster convergence in practice. Similar observations are typically made when comparing the vanilla L-BFGS algorithm with the gradient descent method, one outperforming the other one in many practical cases, but never in theory.

We now move to the proof of the previous results and start by stating few auxiliary results regarding the quality of the gradient approximation provided by Algorithm 1.

4.1 Properties of the gradient approximation

The next lemma is classical (see [17], Lemma 3.1) and provides approximation guarantees about the quantities returned by Algorithm 1. We provide an elementary proof for completeness.

Lemma 1 (Approximation quality of the gradient approximation)**.** *Consider a vector x in $\mathbb{R}^d$, a positive scalar ε and call Algorithm 1:*

$$(g, F_{\mathrm{a}}, z) = \mathsf{GradientEstimate}(x, \varepsilon),$$

[2]Here, $\tilde{O}$ hides logarithmic quantities in μ, L and κ.

where g, F_{a}, z are respectively the approximate gradient, function value, and proximal mapping returned by the algorithm. Then, the following inequalities hold

$$F(x) \leq F_{\mathrm{a}} \leq F(x) + \varepsilon, \tag{14}$$

$$\|z - p(x)\| \leq \sqrt{\frac{2\varepsilon}{\kappa}}, \tag{15}$$

$$\|g - \nabla F(x)\| \leq \sqrt{2\kappa\varepsilon}. \tag{16}$$

Proof. (14) is straightforward by definition of $h(z)$ in Algorithm 1. Since f is convex, the function h is κ-strongly convex, and (15) follows from

$$\frac{\kappa}{2}\|z - p(x)\|^2 \leq h(z) - h(p(x)) \leq \varepsilon,$$

where we recall that $p(x)$ minimizes h. Finally, we also immediately obtain (16) since

$$g - \nabla F(x) = \kappa(x - z) - \kappa(x - p(x)) = \kappa(p(x) - z),$$

by using the definitions of g and the property (5). □

This lemma allow us to quantify the quality of the gradient and function value approximations in terms of ε, which will allow us to control the error accumulation of our algorithm. Before moving to a first convergence result, a few additional technical inequalities are needed.

Lemma 2 (Simple inequalities regarding the gradient approximation)**.** *Consider the same quantities introduced in Lemma 1. Then,*

$$f(z) = F_{\mathrm{a}} - \frac{1}{2\kappa}\|g\|^2, \tag{17}$$

$$\frac{1}{2}\|\nabla F(x)\|^2 - 2\kappa\varepsilon \leq \|g\|^2 \leq 2(\|\nabla F(x)\|^2 + 2\kappa\varepsilon). \tag{18}$$

Proof. Eq. (17) is obtained by simply using the definitions of g and noticing that $F_{\mathrm{a}} = h(z)$. Eq. (18) follows from

$$\begin{aligned} \|\nabla F(x)\|^2 &\leq 2(\|\nabla F(x) - g\|^2 + \|g\|^2) \\ (16) &\leq 2(2\kappa\varepsilon + \|g\|^2). \end{aligned}$$

Interchanging $\nabla F(x)$ and g gives the right-hand side inequality. □

4.2 Convergence analysis of the algorithm (outer-loop)

We are now in shape to establish the convergence of the QuickeNing meta-algorithm, without considering yet the cost of solving the sub-problems (8). Before providing a convergence rate for the sequence $(f(z_k) - f^*)_{k\geq 0}$, we present a result about an approximate descent property. We remark that we have always the relation $(g_k, F_k, z_k) = \mathsf{GradientEstimate}(x_k, \varepsilon_k)$, which allows us to apply Lemma 2 for all $k \geq 0$.

Lemma 3 (Approximate descent property)**.** *Consider the sequence $(x_k)_{k\geq 0}$ generated by Algorithm 3. Then,*

$$\max\{F(x_{k+1}), f(z_k)\} \leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + 3\varepsilon_k. \tag{19}$$

Proof. First, by using successively (17), (14), and (18), we have

$$\begin{aligned} f(z_k) = F_k - \frac{1}{2\kappa}\|g_k\|^2 &\leq F(x_k) + \varepsilon_k - \left(\frac{1}{4\kappa}\|\nabla F(x_k)\|^2 - \varepsilon_k\right) \\ &\leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + 3\varepsilon_k. \end{aligned}$$

Next, to upper bound $F(x_{k+1})$, we consider two cases:

- When the condition (10) is not satisfied, then we set $x_{k+1} = z_k$. As a result,

$$F(x_{k+1}) \leq f(x_{k+1}) = f(z_k) \quad \text{(from (6))}.$$

  Then, the previous inequality on $f(z_k)$ provides the desired result.

- Otherwise, we choose $x_{k+1} = x_{\text{test}}$ and $F_{k+1} = F_{\text{test}}$. Thus,

$$\begin{aligned} F(x_{k+1}) &\leq F_{k+1}, \quad \text{(from (14))} \\ &\leq F_k - \frac{1}{4\kappa}\|g_k\|^2 + \varepsilon_k, \quad \text{(from condition (10))} \\ &\leq F(x_k) + \varepsilon_k - \left(\frac{1}{8\kappa}\|\nabla F(x_k)\|^2 - \frac{1}{2}\varepsilon_k\right) + \varepsilon_k, \quad \text{(from (14) and (18))} \\ &\leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + 3\varepsilon_k. \end{aligned}$$

  This concludes the proof.

□

We may now use the approximate descent property of the previous lemma to control the accumulation of errors in our algorithm, in the strongly convex case.

Proposition 4 (Convergence of Algorithm 3 for strongly convex objectives)**.** *Assume that f is μ-strongly convex and define $\rho = \frac{\mu}{8(\mu+\kappa)}$. Then, the iterates $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ produced by Algorithm 3 satisfy*

$$\max\{F(x_{k+1}) - F^*, f(z_k) - f^*\} \leq (1-2\rho)^{k+1}\left(f(x_0) - f^*\right) + 3\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i.$$

Proof. First, we recall that from Proposition 1, F is strongly convex with parameter $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$ and that the optimal value of the objectives f and F coincide—in other words, and $F^* = f^*$. We may then apply Lemma 3 and

$$\begin{aligned} \max\{F(x_{k+1}) - F^*, f(z_k) - f^*\} &\leq F(x_k) - F^* - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + 3\varepsilon_k, \\ &\leq F(x_k) - F^* - \frac{\mu_F}{4\kappa}(F(x_k) - F^*) + 3\varepsilon_k, \\ &\leq (1-2\rho)\left(F(x_k) - F^*\right) + 3\varepsilon_k, \\ &\leq (1-2\rho)^{k+1}\left(F(x_0) - F^*\right) + 3\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i, \\ &\leq (1-2\rho)^{k+1}\left(f(x_0) - f^*\right) + 3\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i, \end{aligned}$$

where the last inequality simply comes from (6). □

The previous theorem characterizes both the convergence of $(F(x_k))_{k\geq 0}$ and $(f(z_k))_{k\geq 0}$, providing a linear rate as soon as $(\varepsilon_k)_{k\geq 0}$ decreases fast enough. It is then easy to prove the first part of Proposition 2.

Proof of (12). Consider the same quantities introduced in Theorem 4, and choose the sequence

$$\varepsilon_k = \frac{1}{3}C(1-\rho)^{k+1} \quad \text{with} \quad C \geq f(x_0) - f^*.$$

Then, from Proposition 4, we have

$$\begin{aligned}
\max\{F(x_{k+1}) - F^*, f(z_k) - f^*\} &\leq (1-2\rho)^{k+1}\,(f(x_0) - f^*) + 3\sum_{i=0}^{k}(1-2\rho)^{k-i}\,\varepsilon_i,\\
&\leq (1-2\rho)^{k+1}\,C + \sum_{i=0}^{k}(1-2\rho)^{k-i}\,(1-\rho)^{i+1}C,\\
&= C\,\frac{(1-\rho)^{k+2} - (1-2\rho)^{k+2}}{(1-\rho)-(1-2\rho)},\\
&\leq \frac{C}{\rho}(1-\rho)^{k+2}.
\end{aligned}$$

This concludes (12). □

We remark that of course, the quantity $f(x_0) - f^*$ is unknown in practice, but the proposition only requires an upper-bound, e.g., a duality gap, or simply $f(x_0)$ if the function is non-negative. Next, we present the computational complexity analysis, which takes into account both the convergence rate of $(f(z_k))_{k\geq 0}$ and the cost of each iteration of $\mathcal{M}$ used to solve the sub-problems (8).

4.3 Global computational complexity

The complexity analysis of our algorithm uses a few elements from [24] such as the following lemma:

Lemma 4 (ε-accuracy by moving the prox-center ([24], Lemma B.1))**.** *For any vector x in $\mathbb{R}^d$, consider the set*

$$p_\varepsilon(x) \triangleq \{z \in \mathbb{R}^d : h(z) - h^* \leq \varepsilon\} \quad \textit{with} \quad h(z) \triangleq f(z) + \frac{\kappa}{2}\|x - z\|^2,$$

where, as usual, h^* denotes the minimum value of h. Then, if z is in $p_\varepsilon(x)$ for some vector x, we also have $z \in p_{\varepsilon'}(y)$ for any vector y in $\mathbb{R}^d$ with

$$\varepsilon' = 2\varepsilon + \frac{\kappa^2}{\kappa+\mu}\|y - x\|^2.$$

The lemma allows us to control the relations between the different quantities $z_k \in p_{\varepsilon_k}(x_k)$, $z_{k+1} \in p_{\varepsilon_{k+1}}(x_{k+1})$ and $z_{\text{test}} \in p_{\varepsilon_{k+1}}(x_{\text{test}})$ of Algorithm 3. Specifically, we are interested in the relation given in the next lemma:

Lemma 5 (ε-accuracy for the complexity analysis of QuickeNing)**.** *Consider the same assumptions and quantities as in Proposition 2. At iteration k of Algorithm 3, the procedure* GradientEstimate *is called with $x = x_{test}$, and also possibly with $x = x_{k+1} = z_k$. Then, in both cases, z_k is in $p_{\varepsilon'}(x_{test})$ or $p_{\varepsilon'}(z_k)$ with*

$$\varepsilon' = 2\varepsilon_k\left(1 + 2\left(\frac{C}{\rho} + 1\right)\kappa^2 \max\{\lambda_{\max}(H_k)^2, 1/\kappa^2\}\right),$$

where C is the constant introduced in Proposition 3 and $\lambda_{\max}(H_k)$ denotes the largest eigenvalues of H_k.

Proof. In fact, z_k is in $p_{\varepsilon_k}(x_k)$ and

$$\begin{aligned}
\|x_{\text{test}} - x_k\|^2 = \|H_k g_k\|^2 &\leq \lambda_{\max}(H_k)^2\|g_k\|^2\\
&\leq 2\lambda_{\max}(H_k)^2(\|\nabla F(x_k)\|^2 + 2\kappa\varepsilon_k) \quad (\text{ from (18) })\\
&\leq 2\lambda_{\max}(H_k)^2(2\kappa(F(x_k) - F^*) + 2\kappa\varepsilon_k) \quad (\text{Lipschitz gradient})\\
&\leq 4\kappa\lambda_{\max}(H_k)^2\left(\frac{C}{\rho}(1-\rho)^{k+2} + \varepsilon_k\right) \quad (\text{from (12)})\\
&= 4\kappa\left(\frac{C}{\rho} + 1\right)\lambda_{\max}(H_k)^2\varepsilon_k.
\end{aligned}$$

Note that the inequality (12) was proven in the previous section. Applying Lemma 4 with $x = x_k$ and $y = x_{\text{test}}$ shows that $z_k \in p_{\varepsilon'}(x_{\text{test}})$. Replacing x_{test} by $x_{k+1} = z_k$ and noticing that $z_k - x_k = g_k/\kappa$, it suffices to replace H_k by $1/\kappa I_d$ and to follow exactly the same analysis to obtain $z_k \in p_{\varepsilon'}(z_k)$. □

Then, we need to control the largest eigenvalue of H_k thanks to the following lemma, which is a classical step for analyzing the convergence of L-BFGS algorithms [25] [25].

Lemma 6 (Largest eigenvalue of the matrix H_k)**.** *Let H_k be constructed by the modified L-BFGS update presented in Algorithm 2; Then,*

$$\lambda_{\max}(H_k) \leq \frac{1}{\kappa c_1^l} \left(1 + \frac{\kappa}{\mu}\right)^l \left(d + \frac{l}{c_2}\right)^{d+l-1},$$

where c_1, c_2 are constants of modified L-BFGS update in (9), l is the limited memory parameter and d is the dimension of H_k.

Proof. By construction of the L-BFGS rule, H_k is in fact the inverse of the approximate hessian $\tilde{B_k}$, which is constructed from $\tilde{B_0} = \kappa I$ and the generating list $\{(s_i^k, y_i^k)\}_{i=1\ldots j}$ by performing recursively at most l times the BFGS update rule:

$$B' = B + \frac{yy^\top}{y^\top s} - \frac{Bss^\top B}{s^\top Bs},$$

where (s, y) is a pair from the generating list, and B takes the new value B' after each recursion. After scanning all elements of the list and performing $j \leq l$ times the recursion, we obtain $\tilde{B_k}$.

We may now proceed and characterize some constants $\lambda_1, \lambda_2 > 0$ such that

$$\lambda_1 \geq \lambda_{\max}(\tilde{B_k}) \geq \lambda_{\min}(\tilde{B_k}) \geq \lambda_2.$$

Classical analysis shows that with the BFGS update rule, B' is positive definite as soon as B is also positive definite and $s^\top y > 0$. By construction, $s_i^{k\top} y_i^k > 0$ is ensured by (9) for any (s_i^k, y_i^k) in the generating list. Consequently, a simple recurrence shows that all matrices $\tilde{B_k}$ are positive definite and therefore $H_k = {\tilde{B_k}}^{-1}$ are well defined. To show that the eigenvalues of $\tilde{B_k}$ are bounded, we follow in part the classical analysis of [25] by studying the trace and determinants of $\tilde{B_k}$. First,

$$\text{Tr}(B') = \text{Tr}(B) - \frac{\|Bs\|^2}{s^\top Bs} + \frac{\|y\|^2}{y^\top s}.$$

Since the pairs (s, y) from the generating list satisfy $c_2/L_F\|y\|^2 < y^\top s$ from (9),

$$\text{Tr}(B') \leq \text{Tr}(B) + \frac{L_F}{c_2}.$$

Each $\tilde{B_k}$ is obtained by performing at most l times BFGS update from $\tilde{B_0}$ and $L_F = \kappa$, which leads to

$$\text{Tr}(\tilde{B_k}) \leq \text{Tr}(\tilde{B_0}) + \frac{lL_F}{c_2} = \kappa \left(d + \frac{l}{c_2}\right).$$

Since $\tilde{B_k}$ is definite positive, then $\lambda_{\max}(\tilde{B_k}) \leq \text{Tr}(\tilde{B_k}) \leq \lambda_1$ with $\lambda_1 = \kappa\,(d + l/c_2)$. Furthermore, the determinant of B' satisfies the following relation:

$$\det(B') = \det(B)\frac{y^\top s}{s^\top Bs} = \det(B)\,\frac{y^\top s}{s^\top s}\,\frac{s^\top s}{s^\top Bs}.$$

Again, since any pair (s, y) in the generating list satisfies $c_1\mu_F\|s\|^2 < y^\top s$ from (9) and $\lambda_{\max}(B)$ is bounded by λ_1, we obtain

$$\det(B') \geq \det(B) \cdot c_1\mu_F \cdot \frac{1}{\lambda_{\max}(B)} \geq \det(B)\frac{c_1\mu_F}{\lambda_1}.$$

As a result, since $\tilde{B}_k$ is obtained by performing at most l times update, we have

$$\det(\tilde{B}_k) \geq \det(\tilde{B}_0) \left(\frac{c_1\mu_F}{\lambda_1}\right)^l = \kappa^d \left(\frac{c_1\mu_F}{\lambda_1}\right)^l.$$

Therefore,

$$\lambda_{\min}(\tilde{B}_k) \geq \frac{\det(\tilde{B}_k)}{\lambda_{\max}(\tilde{B}_k)^{d-1}} \geq \frac{\det(\tilde{B}_k)}{\lambda_1^{d-1}} \geq \frac{(c_1\mu_F)^l\kappa^d}{\lambda_1^{d+l-1}}.$$

We conclude by remarking that

$$\begin{aligned}\lambda_{\max}(H_k) = \frac{1}{\lambda_{\min}(\tilde{B}_k)} &\leq \frac{\lambda_1^{d+l-1}}{(c_1\mu_F)^l\kappa^d} = \frac{(\kappa(d+\frac{l}{c_2}))^{d+l-1}}{(c_1\frac{\mu\kappa}{\mu+\kappa})^l\kappa^d} \\ &= \frac{1}{\kappa c_1^l}\left(1+\frac{\kappa}{\mu}\right)^l\left(d+\frac{l}{c_2}\right)^{d+l-1}.\end{aligned}$$

□

By combining the two previous lemmas, we see that the quantity ε' from Lemma 5 is proportional to ε_k up to a constant factor which is independent of k. This allows us to conclude the proof of the last part of Proposition 2. Solving the sub-problems (8) up to accuracy $\varepsilon_{k+1} = (1-\rho)\varepsilon_k$ can be done in a constant number of iterations $T_{\mathcal{M}}$ of the method $\mathcal{M}$ according to assumption (11). The only caveat is that $T_{\mathcal{M}}$ increases linearly with the dimension d, which is also a limitation of classical worst-case convergence analysis of L-BFGS methods. The arguments to conclude the proof of Proposition 2 are in fact exactly the same as in Proposition 3.2 of Catalyst [24].

Similarly, Proposition 3 extends the result for the methods SDCA, MISO and Finito, which admit a convergence rate in terms of a dual certificate. The proof requires a small modification that already appears in the proof of Catalyst (see Section D.3 of [24]). In this case, a restart point $z + \frac{\kappa}{\kappa+\mu}(y-x)$ is used to obtain a better certificate at the restart point. More precisely, lemma 4 will hold with $\varepsilon'_{dual} = \varepsilon'/2$ (see Lemma D.5 in [24]). The rest of the proof remains the same as in the primal case.

5 Experiments and practical details

In this section, we present preliminary numerical illustrations and discuss practical matters such as the choice of hyper-parameters.

5.1 Choice of hyper-parameters and variants

The QuickeNing algorithm requires choosing four parameters, one set related to the L-BFGS update, and another one related to the Moreau-Yosida regularization.

Choice of L-BFGS parameters c_1, c_2 and l As discussed previously, the parameters c_1 and c_2 should be intuitively small enough such that the number of skipped L-BFGS steps is reasonable, and large enough to make the matrix H_k well-conditioned. In our experiments, we observed relatively low influence of these parameters; choosing $c_1 = c_2 = 1/2$ seems a safe choice in practice. We used $l = 10$ in our experiments.

Choice of κ and $(\varepsilon_k)_{k\geq 0}$ Like classical L-BFGS algorithms, the theoretical complexity, which we present in Section 4, is a worst case and does not outperform other first-order methods [24]. In Catalyst, the theoretical analysis is used to set up the parameter κ; here, the analysis suffers from a mismatch between theory and practice (as any L-BFGS method), and we recommend instead to set up κ as in Catalyst [24], which seems to provide good results in practice. This choice was used successfully in all our experiments.

Checking the condition $h(z) - h^* \leq \varepsilon$ (or not) Algorithm 1 (GradientEstimate) requires a stopping criterion for the method $\mathcal{M}$. In all our experiments, we used a duality gap, computed once after every pass on the data to ensure that the desired accuracy ε has been reached. Yet, our convergence results suggests another simpler strategy consisting of "guessing" what would be an appropriate constant number $T_{\mathcal{M}}$ of iterations of the method $\mathcal{M}$ to use at every iteration of QuickeNing. This heuristic strategy was evaluated successfully for Catalyst [24], where a variant consisted of using the method $\mathcal{M}$ MISO and force it to perform exactly one pass over the data at every outer-loop iteration. Some preliminary experiments (not reported here) seem to indicate that this heuristic is also effective in the context of QuickeNing.

5.2 Experiments

We now present a few experiments to compare the performance of the following methods:

- **QN-MISO**: QuickeNing applied to the method MISO [26] with all hyper-parameters selected as explained in the previous section.
- **Catalyst-MISO**: Catalyst applied to MISO [26], following exactly [24];
- **SAGA**: the vanilla algorithm of [9], with step-size $\gamma = 1/3L$, as suggested in the author's paper. Including SAGA in the comparison is important since it is the only method we consider that adapts automatically to the unknown true strong convexity parameter of the objective.
- **L-BFGS-MS**: Since implementing L-BFGS line-searches correctly is a bit involved, we use the implementation of Mark Schmidt[3], which has been widely used in other comparisons [42]. We use this method for smooth objectives.

We consider two important machine learning problems. One is the logistic regression formulation with ℓ_2-regularization:

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-b_i \langle a_i, x \rangle}\right) + \frac{\mu}{2}\|x\|^2,$$

where the pairs (b_i, a_i) represent training data with labels b_i in $\{-1, +1\}$ and features points a_i in $\mathbb{R}^d$. The second problem is the non-smooth Elastic-Net formulation of [46]:

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}(b_i - \langle a_i, x \rangle)^2 + \lambda\|x\|_1 + \frac{\mu}{2}\|x\|^2.$$

When the feature vectors a_i are normalized, an upper-bound on the Lipschitz constant can be easily determined with $L_{\text{logistic}} = 1/4$ and $L_{\text{elastic-net}} = 1$. To simulate the ill conditioned problem, we choose the strongly convex parameter $\mu = 1/(100n)$. Note that μ is only a lower-bound on the true strong-convexity parameter, since local strong convexity may be hidden in the loss near the optimum.

Then, we consider two machine learning datasets, presented below

name	covtype	alpha
n	581 012	250 000
d	54	500

Speed comparison results are reported in terms of gradient evaluations, which dominate the cost of all algorithms, are presented in Figure 1. Our conclusions from this first experiment are encouraging: (i) L-BFGS was always significant behind other approaches that exploit the finite sum structure of the objective; (ii) QuickeNing and SAGA perform equally well on alpha, probably due to some hidden strong convexity in the loss, which Catalyst fails to exploit; (iii) for covtype, QuickeNing was significantly faster than other approaches, especially SAGA that seems to suffer from very ill-conditioned problems.

[3]available here `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`

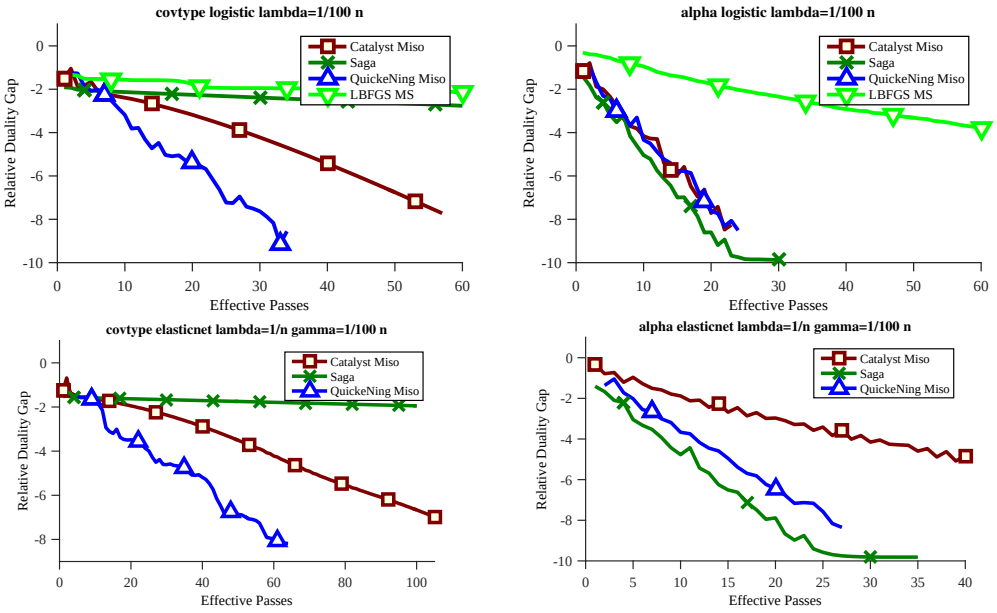

Figure 1: Relative duality gap for different number of passes performed over dataset covtype and alpha. For μ-strongly convex problems, we plot in fact the value $F(x_k)/F^* - 1$ on a logarithmic scale, where F^* is estimated with a duality gap. The legend for all curves is on the top right.

6 Discussions and concluding remarks

A few questions naturally arise regarding the QuickeNing scheme. In particular, one may wonder whether or not our convergence rates may be improved, or whether or not the Moreau-Yosida regularization could be replaced by another smoothing technique. In this section, we discuss these two points and present concluding remarks.

6.1 Discussion of convergence rates

Proposition 2 establishes a linear convergence of QuickeNing for strongly convex objectives when the sequence $(\varepsilon_k)_{k\geq 0}$ is appropriately set. Since QuickeNing uses Quasi-Newton steps, one might have expected a faster convergence rate as several Quasi-Newton algorithms often enjoy [7]. The situation is as follows. Consider the BFGS Quasi-Newton algorithm (without limited memory). As shown in [8], if the sequence $(\varepsilon_k)_{k\geq 0}$ decreases super-linearly, then, it is possible to design a scheme similar to QuickeNing that indeed enjoys a super-linear convergence rate. There are two major downsides though. The scheme of [8] with such fast rate requires performing a line-search on F and a super-linearly decreasing sequence $(\varepsilon_k)_{k\geq 0}$ implies an exponentially growing number of iterations in the inner-loops. These two issues make this approach impractical.

Another potential strategy for obtaining a faster convergence rate consists in interleaving a Nesterov-type extrapolation step in the QuickeNing algorithm. Indeed, the convergence rate of QuickeNing scales linearly in the condition number μ_F/L_F, which suggests that a faster convergence rate could be obtained using a Nesterov-type acceleration scheme. Empirically, we did not observe any benefit of such a strategy, probably because of the pessimistic nature of the convergence rates that are typically obtained for Quasi-Newton approaches based on L-BFGS. Obtaining a linear convergence rate for an L-BFGS algorithm is still an important sanity check, but to the best of our knowledge, the gap in performance between these worst-case rates and practice has always been huge for this class of algorithms.

6.2 Other types of smoothing

Algorithm 1 (GradientEstimate) corresponds to applying the Moreau-Yosida regularization first before computing an estimate g of the gradient, which is a particular instance of infimal convolution smoothing [2]. This family of smoothing operators also encompasses the so-called Nesterov smoothing [2]. Other types of smoothing include randomization techniques [12] or other specific strategies tailored for the objective at hand.

One of the main purposes of the Moreau-Yosida regularization is to provide a better conditioning. As recalled in Proposition 1, the gradient of the Moreau-Yosida-smoothed function F is Lipschitz continuous regardless of whether the original function is continuously differentiable or not. Furthermore, the conditioning of F is improved with respect to the original function, with a condition number depending on the amount of smoothing. As highlighted in [2], this property is also shared by other types of infimal convolution smoothing. Therefore, QuickeNing could potentially be extended to such types of smoothing in place of the Moreau-Yosida regularization. A major advantage of our approach, though, is its outstanding simplicity.

6.3 Concluding remarks

To conclude, we have proposed a generic mechanism, QuickeNing, to accelerate existing first-order optimization algorithms with quasi-Newton-type rules to update a variable metric along the iterations. QuickeNing's main features are the compatibility with composite optimization and its practical performance when combined with incremental approaches. The absence of line-search scheme makes it also easy to implement and use, making it a promising tool for solving large-scale machine learning problems. A few questions remain however open regarding the use of the method in a pure stochastic optimization setting, and the gap in performance between worst-case convergence analysis and practice is significant. We are planning to address the first question about stochastic optimization in future work; the second question is unfortunately difficult and is probably one of the main open question in the literature about L-BFGS methods.

References

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

[2] A. Beck and M. Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[3] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[4] J. Burke and M. Qian. On the superlinear convergence of the variable metric proximal point algorithm using Broyden and BFGS matrix secant updating. *Mathematical Programming*, 88(1):157–181, 2000.

[5] R. Byrd, S. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[6] R. H. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for L-1 regularized optimization. *Mathematical Programming*, 157(2):375–396, 2015.

[7] R. H. Byrd, J. Nocedal, and Y.-X. Yuan. Global convergence of a case of quasi-Newton methods on convex problems. *SIAM Journal on Numerical Analysis*, 24(5):1171–1190, 1987.

[8] X. Chen and M. Fukushima. Proximal quasi-Newton methods for nondifferentiable convex optimization. *Mathematical Programming*, 85(2):313–334, 1999.

[9] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[10] A. Defazio, J. Domke, and T. S. Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2014.

[11] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

[12] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[13] M. Elad. *Sparse and Redundant Representations*. Springer, 2010.

[14] M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

[15] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2015.

[16] M. Fuentes, J. Malick, and C. Lemaréchal. Descentwise inexact proximal algorithms for smooth optimization. *Computational Optimization and Applications*, 53(3):755–769, 2012.

[17] M. Fukushima and L. Qi. A globally and superlinearly convergent algorithm for nonsmooth convex minimization. *SIAM Journal on Optimization*, 6(4):1106–1120, 1996.

[18] R. M. Gower, D. Goldfarb, and P. Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. *preprint arXiv:1603.09649*, 2016.

[19] O. Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

[20] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I.* Springer, 1996.

[21] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms. II.* Springer, 1996.

[22] J. Lee, Y. Sun, and M. Saunders. Proximal Newton-type methods for convex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 836–844, 2012.

[23] C. Lemaréchal and C. Sagastizábal. Practical aspects of the Moreau–Yosida regularization: Theoretical preliminaries. *SIAM Journal on Optimization*, 7(2):367–385, 1997.

[24] H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

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

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

[27] J. Mairal, F. Bach, and J. Ponce. *Sparse Modeling for Image and Vision Processing*. Foundations and Trends in Computer Graphics and Vision. 2014.

[28] R. Mifflin. A quasi-second-order proximal bundle algorithm. *Mathematical Programming*, 73(1):51–72, 1996.

[29] P. Moritz, R. Nishihara, and M. I. Jordan. A linearly-convergent stochastic L-BFGS algorithm. *preprint arXiv:1508.02087*, 2015.

[30] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[31] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

[32] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[33] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[34] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[35] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[36] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[37] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[38] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[39] S. Salzo and S. Villa. Inexact and accelerated proximal point algorithms. *Journal of Convex Analysis*, 19(4):1167–1192, 2012.

[40] K. Scheinberg and X. Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. *Mathematical Programming*, pages 1–35, 2014.

[41] M. Schmidt, D. Kim, and S. Sra. *Projected Newton-type methods in machine learning*, pages 305–330. MIT Press, 2011.

[42] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2016.

[43] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 2015.

[44] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[45] J. Yu, S. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to non-smooth convex optimization. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2008.

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