An Inexact Variable Metric Proximal Point Algorithm for Generic Quasi-Newton Acceleration

Hongzhou Lin, Julien Mairal, Zaid Harchaoui

An Inexact Variable Metric Proximal Point Algorithm for Generic Quasi-Newton Acceleration

Hongzhou Lin
MIT*
hongzhou.lin@inria.fr

Julien Mairal
Inria†
julien.mairal@inria.fr

Zaid Harchaoui
University of Washington‡
zaid@uw.edu

July 20, 2018

Abstract

We propose an inexact variable-metric proximal point algorithm to accelerate gradient-based optimization algorithms. The proposed scheme, called QNing, can be notably applied to incremental first-order methods such as the stochastic variance-reduced gradient descent algorithm (SVRG) and other randomized incremental optimization algorithms. QNing 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. When combined with limited-memory BFGS rules, QNing is particularly effective to solve high-dimensional optimization problems, while enjoying a worst-case linear convergence rate for strongly convex problems. We present experimental results where QNing gives significant improvements over competing methods for training machine learning methods on large samples and in high dimensions.

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. A typical example from the signal and image processing literature is the ℓ_1-norm $\psi(x) = \|x\|_1$, which encourages sparse solutions [19, 40]; 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 [28]). 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

*Computer Science and Artificial Intelligence Laboratory Cambridge, MA 02139, USA. This work was performed in large parts while Hongzhou Lin was at Inria.
†Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, Grenoble, 38000, France.
‡Department of Statistics Seattle, WA 98195, USA.

independent terms? Is ψ separable in different blocks of coordinates? (iii) exploiting the local curvature of the smooth term f to achieve faster convergence than gradient-based approaches when the dimension d is large. Typically, the first point is well understood in the context of optimal first-order methods, see [2, 48], and the third point is tackled with effective heuristics such as L-BFGS when the problem is smooth [35, 49]. Yet, tackling all these challenges at the same time is difficult, which is precisely the focus of this paper.

In particular, a problem of interest that initially 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 [56], SAGA [15], SDCA [58], SVRG [60], Finito [16], or MISO [38]. 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 [2, 48] 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 [21, 33, 58].

To tackle (2), dedicated first-order methods are often the default choice in machine learning, but it is also known that standard Quasi-Newton approaches can sometimes be surprisingly effective in the smooth case—that is when $\psi = 0$, see, e.g., [56] 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 [35, 49]. The theoretical guarantees offered by L-BFGS are somewhat limited, 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, L-BFGS remains one of the greatest practical success of smooth optimization. Adapting L-BFGS to composite and structured problems, such as the finite sum of functions (2), is of utmost importance nowadays.

For instance, there have been several attempts to develop a proximal Quasi-Newton method [10, 31, 54, 62]. These algorithms typically require computing many times the proximal operator of ψ with respect to a variable metric. Quasi-Newton steps were also incorporated as local search steps into accelerated first-order methods to further enhance their numerical performance [24]. More related to our work, L-BFGS is combined with SVRG for minimizing smooth finite sums in [26]. The scope of our approach is broader beyond the case of SVRG. We present a generic Quasi-Newton scheme, applicable to a large-class of first-order methods for composite optimization, including other incremental algorithms [15, 16, 38, 56, 58] and block coordinate descent methods [51, 52]

More precisely, the main contribution of this paper is a generic meta-algorithm, called QNing (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 appropriate accuracy, resulting in faster global convergence in practice. QNing falls into the class of inexact proximal point algorithms with variable metric and *may be seen as applying a Quasi-Newton 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) when used with L-BFGS rules, it admits a worst-case linear convergence rate for strongly convex problems similar to that of gradient descent, which is typically the best guarantees obtained for L-BFGS schemes in the literature.

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 [14, 23, 41], 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 *a simple line search scheme*, and with *warm start strategies for the*

sub-problems 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 QNing on regularized logistic regression and regularized least-squares, with smooth and nonsmooth regularization penalities such as the Elastic-Net [63]. We use large-scale machine learning datasets and show that QNing performs at least as well as the recently proposed Catalyst [33] and as the classical L-BFGS scheme in all numerical experiments, and significantly outperforms them in many cases.

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

2 Related work and preliminaries

The history of quasi-Newton methods can be traced back to the 1950's [6, 29, 50]. Quasi-Newton methods often lead to significantly faster convergence in practice compared to simpler gradient-based methods for solving smooth optimization problems [55]. 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 briefly review the well-known BFGS algorithm in Section 2.1, its limited memory variant [49], and a few recent extensions. Then, we present earlier works that combine proximal point and Quasi-Newton algorithms in Section 2.3.

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 [50]. These approaches will be the workhorses of the QNing meta-algorithm in practice. Consider a smooth convex objective f to be minimized, the BFGS method constructs at iteration k 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 $y_k^\top s_k > 0$ and the positive definiteness of B_k are guaranteed as soon as f is strongly convex. To determine the stepsize α_k, Wolfe's line-search is a simple choice which provides linear convergence rate in the worst case. In addition, if the objective is twice differentiable and the Hessian is Lipschitz continuous, the convergence is asymptotically superlinear [50].

The limited memory variant L-BFGS [49] overcomes the issue of storing B_k for large d, by replacing it by another positive definite matrix—say $\bar{B}_k$—which can be built from a "generating list" of at most l pairs of vectors $\{(s_i^k, y_i^k)\}_{i=0\dots j}$ along with an initial diagonal matrix $\bar{B}_0$. Formally, $\bar{B}_k$ can be computed by applying at most l times a recursion similar to (3) involving all 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 (when $j = l$) and adding a new one. What makes the approach appealing is the ability of computing $H_k z = B_k^{-1} z$ 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) [57, 42, 8]. Unfortunately substituting the full gradient $\nabla f(x_k)$ by its stochastic counterpart does not lead to a convergent scheme. Instead, the SQN method [8] 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 [26, 44].

2.2 Quasi-Newton methods for composite optimization

Different approaches have been proposed to extend Quasi-Newton methods to composite optimization problems. A first approach consists in minimizing successive quadratic approximations, also called proximal quasi-Newton methods [10, 25, 30, 31, 36, 54]. More concretely, a quadratic approximation q_k is minimized at each iteration:

$$q_k(x) \triangleq f_0(x_k) + \langle \nabla f_0(x_k), x - x_k \rangle + \frac{1}{2}(x - x_k)^T B_k (x - x_k) + \psi(x), \tag{4}$$

where B_k is a Hessian approximation based on quasi-Newton methods. The minimizer of q_k provides a descent direction, which is subsequently used to build the next iterate. However, since B_k is dense and changes over the iterations, a closed form solution of (4) is usually not available, and one needs to apply an optimization algorithm to approximately solve (4). Even though local superlinear convergence may be guaranteed under mild assumptions when (4) is solved with "high accuracy" [31], the composite structure naturally leads to choosing a first-order algorithm for solving (4). Then, superlinear complexity becomes out of reach. The global convergence rate of this inexact variant has been for instance analyzed in [54], where a sublinear convergence rate is obtained for convex problems by using a randomized coordinate descent solver applied to (4); later, a linear convergence rate was obtained by [36] for strongly convex problems.

A second approach to extend Quasi-Newton methods to composite optimization problems is based on a smoothing technique. More precisely, any Quasi-Newton method may be applied to a smoothed version of the objective. For instance, one may use the forward-backward envelope [4, 59] to build forward-backward quasi Newton methods. The idea is to mimic forward-backward splitting methods and apply quasi-Newton methods instead of gradient methods on top of the envelope. Another well known smoothing technique is the Moreau-Yosida regularization [43, 61], which leads to the variable metric proximal point algorithm [7, 14, 22, 23]. Our method pursues this line of work by developing a practical inexact variant with global complexity guarantees.

2.3 Combining the proximal point algorithm and Quasi-Newton methods

We briefly recall the definition of the Moreau-Yosida regularization and its basic properties.

Definition 1. *Given an objective function f and a smoothing parameter $\kappa > 0$, the Moreau-Yosida regularization of f is defined as the infimal convolution*

$$F(x) \triangleq \min_{z \in \mathbb{R}^d} \left\{ f(z) + \frac{\kappa}{2}\|z - x\|^2 \right\}. \tag{5}$$

When f is convex, the sub-problem defined in (5) is strongly convex which provides an unique minimizer, called the proximal point of x, which we denote by $p(x)$.

Proposition 1 (Basic properties of the Moreau-Yosida regularization). *If f is convex, the Moreau-Yosida regularization F defined in (5) satisfies*

1. *Minimizing f and F are equivalent in the sense that*

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

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

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

   $$\nabla F(x) = \kappa(x - p(x)). \tag{6}$$

   Moreover the gradient ∇F is Lipschitz continuous with constant $L_F = \kappa$.

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

Interestingly, F inherits all the convex properties of f and more importantly it is always continuously differentiable, see [32] for elementary proofs. Moreover, the condition number of F is given by

$$q = \frac{L_F}{\mu_F} = \frac{\mu + \kappa}{\mu}, \tag{7}$$

which is driven by the regularization parameter κ. Naturally, a naive approach for minimizing a possibly non-smooth function f is to apply an optimization method on F since both functions admit the same solutions. This yields the following well-known algorithms.

The proximal point algorithm. Consider gradient descent 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 the proximal point algorithm [53]:

$$x_{k+1} = p(x_k) = \operatorname*{arg\,min}_{z \in \mathbb{R}^d} \left\{ f(z) + \frac{\kappa}{2}\|z - x_k\|^2 \right\}. \tag{8}$$

Accelerated 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 [45] uses a two-stage update, using a specific extrapolation parameter β_{k+1}:

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

and, given (6), we obtain that $x_{k+1} = p(y_k)$. This is known as the accelerated proximal point algorithm introduced by Güler [27], which was recently extended in [33, 34].

Variable metric proximal point algorithm. One can also apply Quasi-Newton methods on top of the Moreau-Yosida regularization, which yields

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla F(x_k), \tag{9}$$

where B_k is the Hessian approximation based on Quasi-Newton methods. This is known as the variable metric proximal point algorithm [7, 14, 22, 23].

Towards an inexact variable metric proximal point algorithm. Quasi-Newton approaches have been applied after inexact Moreau-Yosida smoothing in various ways [7, 14, 22, 23]. In particular, it is shown in [14] that if the sub-problems (5) are solved up to high enough accuracy, then the inexact variable metric proximal point algorithm preserves the superlinear convergence rate. However, the complexity for solving the sub-problems with high accuracy is typically not taken into account in such previous work. The main contribution of our paper is to close this gap by providing a global analysis and algorithmic choices allowing to use a first-order method in the inner-loop. More precisely, in the proposed QNing algorithm, we provide i) a simple line-search strategy which guarantees sufficient descent in terms of function value; ii) a practical stopping criterion for the sub-problems; iii) several warm-start strategies. These three components together yields the global convergence analysis which takes into account the inner-loop complexity.

Explicit vs. implicit gradient methods. The classical Quasi-Newton rule (3) and the variable metric proximal point update (9) are related since they only differ by the point chosen to evaluate the gradient of f. The first rule performs indeed an explicit gradient step $x_{k+1} = x_k - \alpha_k B_k^{-1}\nabla f(x_k)$, whereas it is possible to show that (9) is equivalent to $x_{k+1} = x_k - \alpha_k B_k^{-1}\nabla f(z_k)$, where $z_k = p(x_k)$. The latter is often referred to as an implicit gradient step, since the point z_k is not known in advance and requires solving a sub-problem.

In the unrealistic case where $p(x_k)$ can be obtained at no cost, implicit gradient steps can afford much larger step sizes than explicit ones and are more effective. For instance, when f is strongly convex, by choosing $\alpha_k B_k^{-1} = 1/\kappa$, it is possible to get arbitrarily close to the optimum in a single gradient step by making κ arbitrarily small. In practice, however, sub-problems are solved only approximately, and whether or not one should prefer explicit or inexact implicit steps is less clear. A small κ makes the smoothed function F better conditioned, while a large κ is needed to improve the conditioning of the sub-problem (5).

In the composite case, both approaches require approximately solving sub-problems, namely (4) and (5), respectively. In the general case, when a generic first-order method—*e.g.*, proximal gradient descent—is used, our worst-case complexity analysis does not provide a clear winner, and our experiments in Section 5.4 confirm that both approaches perform similarly. However, when it is possible to exploit the specific structure of the sub-problems in one case, but not in the other one, the conclusion may differ.

For instance, the implicit strategy applied to a finite sum (2) leads to sub-problems that can be solved in $O(n \log(1/\varepsilon))$ iterations with SVRG [60], SAGA [15] or MISO [38], by using the same choice $\kappa = 2L/n$ as Catalyst [34]. Assuming that computing a gradient of a function f_i and computing the proximal operator of ψ are both feasible in $O(d)$ floating point operations, our approach solves each sub-problem with enough accuracy in $\tilde{O}(nd)$ operations.[1] On the other hand, we cannot naively apply SVRG to solve the proximal Quasi-Newton update (4) at the same cost: (i) assuming that B_k has rank l, computing a single gradient of a sum's component will cost $O(dl)$, resulting in l-fold increase per iteration in terms of computational complexity; (ii) the previous iteration-complexity $O(n \log(1/\varepsilon))$ for solving the sub-problems would require the condition $B_k \succeq (L/n)I$, forcing the Quasi-Newton metric to be potentially more isotropic. For this reason, existing attempts to combine SVRG with Quasi-Newton principles have adopted other directions [26, 44].

3 QNing: a Quasi-Newton meta-algorithm

We now present the QNing method in Algorithm 1, which consists of applying variable metric algorithms on the smoothed objective F with inexact gradients. Each gradient approximation is the result of a minimization problem tackled with the algorithm $\mathcal{M}$, used as a sub-routine. The outer loop of the algorithm performs quasi-Newton updates. The method $\mathcal{M}$ can be any algorithm of the user's choice, as long as it enjoys linear convergence rate for strongly convex problems. More details about the choice of the parameter κ and about the inexactness criterion to use will be given next.

3.1 The main algorithm

We now discuss the main algorithm components and its main features.

Outer-loop: variable metric inexact proximal point algorithm. We apply variable metric algorithms with a simple line search strategy similar to [54] on the Moreau-Yosida regularization. Given a positive definite matrix H_k and a step size η_k in $[0, 1]$, the algorithm computes the update

$$x_{k+1} = x_k - (\eta_k H_k + (1 - \eta_k) H_0) g_k, \tag{LS}$$

where $H_0 = \kappa^{-1} I$. When $\eta_k = 1$, the update uses the metric H_k, and when $\eta_k = 0$, it uses an inexact proximal point update $x_{k+1} = x_k - (1/\kappa) g_k$. In other words, when the quality of the metric H_k is not good enough, due to the inexactness of the gradients used for its construction, the update is corrected towards that of a simple proximal point update, whose convergence is well understood when the gradients are inexact.

In order to choose the stepsize, we introduce the following descent condition,

$$F_{k+1} \leq F_k - \frac{1}{4\kappa} \|g_k\|^2. \tag{12}$$

[1] The notation $\tilde{O}$ hides logarithmic quantities.

Algorithm 1 QNing: a Quasi-Newton meta-algorithm

input Initial point x_0 in $\mathbb{R}^d$; number of iterations K; smoothing parameter $\kappa > 0$; optimization algorithm $\mathcal{M}$; optionally, budget $T_{\mathcal{M}}$ for solving the sub-problems.

Initialization: $(g_0, F_0, z_0) = \mathsf{ApproxGradient}\,(x_0, \mathcal{M})$; $H_0 = \kappa^{-1} I$.
for $k = 0, \ldots, K-1$ **do**
  Initialize $\eta_k = 1$.
  Perform the Quasi-Newton step

$$x_{\text{test}} = x_k - (\eta_k H_k + (1 - \eta_k) H_0)\, g_k.$$

  Estimate the gradient and function of the Approximate Moreau envelope at x_{test}

$$(g_{\text{test}}, F_{\text{test}}, z_{\text{test}}) = \mathsf{ApproxGradient}\,(x_{\text{test}}, \mathcal{M})\ .$$

  while $F_{\text{test}} > F_k - \frac{1}{4\kappa}\|g_k\|^2$ **do**
    Decrease the value of the line search parameter η_k in $[0, 1]$ and re-evaluate x_{test}.
    Re-evaluate $(g_{\text{test}}, F_{\text{test}}, z_{\text{test}}) = \mathsf{ApproxGradient}\,(x_{\text{test}}, \mathcal{M})$.
  end while
  Accept the new iterate: $(x_{k+1}, g_{k+1}, F_{k+1}, z_{k+1}) = (x_{\text{test}}, g_{\text{test}}, F_{\text{test}}, z_{\text{test}})$.
  Update H_{k+1} (for example, use $\mathsf{L\text{-}BFGS}(H_k, x_{k+1} - x_k, g_{k+1} - g_k)$).
end for

output inexact proximal point z_K (solution).

Algorithm 2 Generic procedure ApproxGradient

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

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

$$z \approx \arg\min_{w \in \mathbb{R}^d} \left\{ h(w) \triangleq f(w) + \frac{\kappa}{2}\|w - x\|^2 \right\}, \tag{10}$$

using one of the following stopping criteria:

- Stop when the approximate solution z satisfies

$$h(z) - h^* \leq \frac{\kappa}{72}\|z - x\|^2. \tag{11}$$

- Simply use a pre-defined constant budget $T_{\mathcal{M}}$ (for instance one pass over the data).

Estimate the gradient $\nabla F(x)$ of the Moreau-Yosida objective function

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

output approximate gradient estimate g, objective value $F_a \triangleq h(z)$, proximal mapping z.

In our experiments, we observed empirically that the stepsize $\eta_k = 1$ was almost always selected. In practice, we try the values η_k in $\{1, 1/2, 1/4, 1/8, 0\}$ starting from the largest one and stopping whenever condition (12) is satisfied, which can be shown to be the case for $\eta_k = 0$.

Example of variable metric: inexact L-BFGS method. The L-BFGS rule we consider is the standard one and consists in updating incrementally a generating list of vectors $\{(s_i, y_i)\}_{i=1\dots j}$, which implicitly defines the L-BFGS matrix. We use here the two-loop recursion detailed in [50, Algorithm 7.4] and use skipping steps when the condition $s_i^\top y_i > 0$ is not satisfied, in order to ensure the positive-definiteness of the L-BFGS matrix H_k (see [20]).

Inner-loop: approximate Moreau-envelope. The inexactness of our scheme comes from the approximation of the Moreau-envelope F where a minimization algorithm $\mathcal{M}$ is used. The procedure ApproxGradient () calls the minimization algorithm $\mathcal{M}$ to minimize the sub-problem (10). When the problem is solved exactly, the function returns the exact values $g = \nabla F(x)$, $F_z = F(x)$, and $z = p(x)$. However, this is infeasible in practice and we can only expect approximate solutions. In particular, a stopping criterion should be specified. We consider the following variants:

(a) we define an adaptive stopping criterion based on function values and stop $\mathcal{M}$ when the approximate solution satisfies the inequality (11). In contrast to standard stopping criterion where the accuracy is an absolute constant, our stopping criterion is adaptive since the righthand side of (11) also depends on the current iterate z. More detailed theoretical insights will be given in Section 4. Typically, checking whether or not the criterion is satisfied requires computing a duality gap, as in Catalyst [34].

(b) using a pre-defined budget $T_{\mathcal{M}}$ in terms of number of iterations of the method $\mathcal{M}$, where $T_{\mathcal{M}}$ is a constant independent of k.

As we will see later in Section 4, when $T_{\mathcal{M}}$ is large enough, criterion (11) is guaranteed. Note that such an adaptive stopping criterion is relatively classical in the literature of inexact gradient-based methods [9].

Requirements on $\mathcal{M}$. To apply QNing, the optimization method $\mathcal{M}$ needs to have linear convergence rates for strongly-convex problems. More precisely, for any any strongly-convex objective h, the method $\mathcal{M}$ should be able to generate a sequence of iterates $(w_t)_{t\geq 0}$ such that

$$h(w_t) - h^* \leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^t (h(w_0) - h^*) \quad \text{for some constants} \quad C_{\mathcal{M}} > 0 \text{ and } 1 > \tau_{\mathcal{M}} > 0, \tag{13}$$

where w_0 is the initial point given to $\mathcal{M}$. The notion of linearly convergent methods extends naturally to non-deterministic methods where (13) is satisfied in expectation:

$$\mathbb{E}[h(w_t) - h^*] \leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^t (h(w_0) - h^*). \tag{14}$$

The linear convergence condition typically holds for many primal gradient-based optimization techniques, including classical full gradient descent methods, block-coordinate descent algorithms [47, 52], or variance reduced incremental algorithms [15, 56, 60]. In particular, our method provides a generic way to combine incremental algorithms with Quasi-Newton methods which are suitable for large scale optimization problems. For the simplicity of the presentation, we only consider the deterministic variant (13) in the analysis. However, it is possible to show that the same complexity results still hold for non-deterministic methods in expectation, as discussed in Section 4.5. We emphasize that we do not assume any convergence guarantee of $\mathcal{M}$ on non-strongly convex problems since we will always apply $\mathcal{M}$ to strongly convex sub-problems.

Warm starts for the sub-problems. Using the right starting point for initializing the method $\mathcal{M}$ when solving each sub-problem is important to guarantee that the accuracy to ensure global convergence of the

algorithm can be achieved with a constant number of iterations. We show that it is indeed the case with the following choices: Consider the minimization of a sub-problem

$$\min_{w\in\mathbb{R}^d}\left\{h(w)\triangleq f(w)+\frac{\kappa}{2}\|w-x\|^2\right\}.$$

Then, our warm start strategy depends on the nature of f:

- when f is smooth, we use $w_0=x$;
- when $f=f_0+\psi$ is composite, we use

$$w_0=\operatorname*{arg\,min}_{w\in\mathbb{R}^d}\left\{f_0(x)+\langle\nabla f_0(x),w-x\rangle+\frac{L+\kappa}{2}\|w-x\|^2+\psi(w)\right\}.$$

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 characteristics; in particular, the ℓ_1-norm is known to provide sparsity. Smoothing techniques [46] 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 statement, 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 [2] are often preferred (see [40]).

Then, one might be concerned that our scheme operates on the smoothed objective F, leading to iterates $(x_k)_{k\geq 0}$ that may 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 $\mathcal{M}$ 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^*.

4 Convergence and complexity analysis

In this section, we study the convergence of the QNing algorithm—that is, the rate of convergence of the quantities $(F(x_k)-F^*)_{k\geq 0}$ and $(f(z_k)-f^*)_{k\geq 0}$, and also the computational complexity due to solving the sub-problems (10). We start by stating the main properties of the gradient approximation in Section 4.1. Then, we analyze the convergence of the outer loop algorithm in Section 4.2, and Section 4.3 is devoted to the properties of the line search strategy. After that, we provide the cost of solving the sub-problems in Section 4.4 and derive the global complexity analysis in Section 4.5.

4.1 Properties of the gradient approximation

The next lemma is classical and provides approximation guarantees about the quantities returned by the ApproxGradient procedure (Algorithm 2); see [5, 23]. We recall here the proof for completeness.

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

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

such that $h(z) - h^* \leq \varepsilon$, *where* $h^* = \min_{v \in \mathbb{R}^d} h(v)$. *As in Algorithm 2, define* $g = \kappa(x - z)$ *and* $F_a = h(z)$. *Then, the following inequalities hold*

$$F(x) \leq F_{\mathrm{a}} \leq F(x) + \varepsilon, \tag{15}$$

$$\|z - p(x)\| \leq \sqrt{\frac{2\varepsilon}{\kappa}}, \tag{16}$$

$$\|g - \nabla F(x)\| \leq \sqrt{2\kappa\varepsilon}. \tag{17}$$

Moreover, F_a *is related to* f *by the following relationship*

$$f(z) = F_{\mathrm{a}} - \frac{1}{2\kappa}\|g\|^2. \tag{18}$$

Proof. (15) and (18) are straightforward by definition of $h(z)$. Since f is convex, the function h is κ-strongly convex, and (16) follows from

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

where we recall that $p(x)$ minimizes h. Finally, we obtain (17) from

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

by using the definitions of g and the property (6). □

This lemma allows us to quantify the quality of the gradient and function value approximations, which is crucial to control the error accumulation of inexact proximal point methods. Moreover, the relation (18) establishes a link between the approximate function value of F and the function value of the original objective f; as a consequence, it is possible to relate the convergence rate of f from the convergence rate of F. Finally, the following result is a direct consequence of Lemma 1:

Lemma 2 (Bounding the exact gradient by its approximation). *Consider the same quantities introduced in Lemma 1. Then,*

$$\frac{1}{2}\|g\|^2 - 2\kappa\varepsilon \leq \|\nabla F(x)\|^2 \leq 2(\|g\|^2 + 2\kappa\varepsilon). \tag{19}$$

Proof. The right-hand side of Eq. (19) follows from

$$\begin{aligned} \|\nabla F(x)\|^2 &\leq 2(\|\nabla F(x) - g\|^2 + \|g\|^2) \\ &\leq 2(2\kappa\varepsilon + \|g\|^2) \qquad \text{(from (17))}. \end{aligned}$$

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

Corollary 1. *If* $\varepsilon \leq \frac{c}{\kappa}\|g\|^2$ *with* $c < \frac{1}{4}$, *then*

$$\frac{1 - 4c}{2} \leq \frac{\|\nabla F(x)\|^2}{\|g\|^2} \leq 2(1 + 2c). \tag{20}$$

This corollary is important since it allows to replace the unknown exact gradient $\|\nabla F(x)\|$ by its approximation $\|g\|$, at the cost of a constant factor, as long as the condition $\varepsilon \leq \frac{c}{\kappa}\|g\|^2$ is satisfied.

4.2 Convergence analysis of the outer loop

We are now in shape to establish the convergence of the QNing meta-algorithm, without considering yet the cost of solving the sub-problems (10). At iteration k, an approximate proximal point is evaluated:

$$(g_k, F_k, z_k) = \mathsf{ApproxGradient}(x_k, \mathcal{M}) \ . \tag{21}$$

The following lemma characterizes the expected descent in terms of objective function value when the gradient is approximately computed.

Lemma 3 (Approximate descent property). *At iteration k, if the sub-problem (21) is solved up to accuracy ε_k in the sense of Lemma 1 and the next iterate x_{k+1} satisfies the descent condition (12), then,*

$$F(x_{k+1}) \leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + \frac{3}{2}\varepsilon_k. \tag{22}$$

Proof. From (15) and (12),

$$\begin{aligned} F(x_{k+1}) &\leq F_{k+1} \leq F_k - \frac{1}{4\kappa}\|g_k\|^2 \\ &\leq F(x_k) + \varepsilon_k - \left(\frac{1}{8\kappa}\|\nabla F(x_k)\|^2 - \frac{\varepsilon_k}{2}\right) \quad \text{(from (15) and (19))} \\ &= F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 + \frac{3}{2}\varepsilon_k. \end{aligned}$$

□

This lemma gives us a first intuition about the natural choice of the accuracy ε_k, which should be in the same order as $\|\nabla F(x_k)\|^2$. In particular, if

$$\varepsilon_k \leq \frac{1}{16\kappa}\|\nabla F(x_k)\|^2, \tag{23}$$

then we have

$$F(x_{k+1}) \leq F(x_k) - \frac{1}{32\kappa}\|\nabla F(x_k)\|^2, \tag{24}$$

which is a typical inequality used for analyzing gradient descent methods. Before presenting the convergence result, we remark that condition (23) cannot be used directly since it requires knowing the exact gradient $\|\nabla F(x_k)\|$. A more practical choice consists of replacing it by the approximate gradient.

Lemma 4 (Practical choice of ε_k). *The following condition implies inequality (23):*

$$\varepsilon_k \leq \frac{1}{72\kappa}\|g_k\|^2. \tag{25}$$

Proof. From Corollary 1, Equation (25) implies

$$\|g_k\|^2 \leq \frac{9}{2}\|\nabla F(x_k)\|^2 \quad \text{and thus} \quad \varepsilon_k \leq \frac{1}{72\kappa}\|g_k\|^2 \leq \frac{1}{16\kappa}\|\nabla F(x_k)\|^2.$$

□

Whereas the gradient $\|\nabla F(x_k)\|$ is unknown in practice, we have access to the estimate $g_k = \kappa(x_k - z_k)$, which allows us to use condition (25). Finally, we obtain the following convergence result for strongly convex problems, which is relatively classical in the literature of inexact gradient methods (see Section 4.1 of [9] for a similar result).

Proposition 2 (**Convergence of Algorithm 1, strongly-convex objectives).** *Assume that f is μ-strongly convex. Let $(x_k)_{k\geq 0}$ be the sequences generated by Algorithm 1 where the stopping criterion (11) is used. Then,*

$$F(x_k) - F^* \leq \left(1 - \frac{1}{16q}\right)^k (F(x_0) - F^*), \quad \textit{with} \quad q = \frac{\mu + \kappa}{\mu}.$$

Proof. The proof follows directly from (24) and the standard analysis of the gradient descent algorithm for the μ_F-strongly convex and L_F-smooth function F by remarking that $L_F = \kappa$ and $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$. □

Corollary 2. *Under the conditions of Proposition 2, we have*

$$f(z_k) - f^* \leq \left(1 - \frac{1}{16q}\right)^k (f(x_0) - f^*). \tag{26}$$

Proof. From (18) and (25), we have

$$f(z_k) = F_k - \frac{1}{2\kappa}\|g_k\|^2 \leq F(x_k) + \varepsilon_k - \frac{1}{2\kappa}\|g_k\|^2 \leq F(x_k).$$

Moreover, $F(x_0) = \min_{w\in\mathbb{R}^d}\{f(w) + \frac{\kappa}{2}\|w - x_0\|^2\} \leq f(x_0) + \frac{\kappa}{2}\|x_0 - x_0\|^2 = f(x_0)$. □

It is worth pointing out that our analysis establishes a linear convergence rate whereas one would expect a superlinear convergence rate as for classical classical variable metric methods with infinite memory. The tradeoff lies in the choice of the accuracy ε_k. In order to achieve superlinear convergence rate, the approximation error ε_k also needs to decrease superlinearly as shown in [14]. However, a fast decreasing sequence ε_k requires an increasing effort in solving the sub-problems, which will dominate the global complexity. In other words, the global complexity may become worse even though we achieve faster convergence in the outer-loop. This will become clearer when we discuss the inner loop complexity in Section 4.4.

Next, we show the classical sublinear $O(1/k)$ convergence rate of QNing under a bounded level set condition, when the objective is convex but not necessarily strongly convex.

Proposition 3 (**Convergence of Algorithm 1 for convex, but not strongly-convex objectives).** *Let f be a convex function with bounded level sets. Then, there exists a constant $R > 0$, which depends on x_0, such that the sequences $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ generated by Algorithm 1 with stopping criterion (11), satisfies*

$$F(x_k) - F^* \leq \frac{8\kappa R^2}{k} \quad \textit{and} \quad f(z_k) - f^* \leq \frac{8\kappa R^2}{k}.$$

Proof. We defer the proof and the proper definition of the bounded level set assumption to Appendix A. □

So far, the analysis has assumed that the line search would always produce an iterate that satisfies the descent condition (12), which naturally holds for a step size $\eta_k = 0$. In the next section, we study classical conditions under which a non-zero step size is selected.

4.3 Conditions for non-zero step sizes η_k and termination of the line search

At iteration k, the line search is performed on the stepsize η_k to find the next iterate

$$x_{k+1} = x_k - (\eta_k H_k + (1 - \eta_k)H_0)g_k,$$

such that x_{k+1} satisfies the descent condition (12). Intuitively, when η_k goes to zero, x_{k+1} will be close to the classical gradient step where the descent condition holds. This observation leads us to consider the following sufficient condition for the descent condition (12).

Lemma 5 (A sufficient condition for the descent condition (12)). *If $x_{k+1} = x_k - A_k g_k$ where A_k is a positive definite matrix such that $\frac{1-\alpha}{\kappa} I \preceq A_k \preceq \frac{1+\alpha}{\kappa} I$ with $\alpha \leq \frac{1}{32}$ and the sub-problems are solved up to accuracy $\varepsilon_k \leq \frac{1}{72\kappa}\|g_k\|^2$, then the sufficient condition (12) holds, i.e.*

$$F_{k+1} \leq F_k - \frac{1}{4\kappa}\|g_k\|^2. \tag{27}$$

Therefore, a line search strategy consisting of finding the largest η_k of the form γ^i, with $i = 1, \ldots, +\infty$ and γ in $(0, 1)$ always terminates in a bounded number of iterations if the sequence $(H_k)_{k\geq 0}$ is also bounded, meaning there exists $0 < m < M$ such that for any k, $mI \preceq H_k \preceq MI$. Note that in practice, we consider a set of step sizes $\eta_k = \gamma^i$ for $i \leq i_{\max}$ or $\eta_k = 0$, which naturally upper-bounds the number of line search iterations to $i_{\max}$. More precisely, all experiments performed in this paper use $\gamma = 1/2$ and $i_{\max} = 3$.

Proof of Lemma 5. First, we recall that $z_k = x_k - \frac{1}{\kappa} g_k$ and

$$F(z_k) \leq f(z_k) = F_k - \frac{1}{2\kappa}\|g_k\|^2, \tag{28}$$

which means the step size $\eta_k = 0$ naturally satisfies the descent condition. Considering now $\eta_k \neq 0$, we have,

$$F_{k+1} = \underbrace{F_{k+1} - F(x_{k+1})}_{\triangleq E_1} + \underbrace{F(x_{k+1}) - F(z_k)}_{\triangleq E_2} + F(z_k),$$

and we are going to bound the two error terms E_1 and E_2 by some factors of $\|g_k\|^2$.

By denoting $\varepsilon_k \leq \frac{c}{\kappa}\|g_k\|^2$ with $c = \frac{1}{72}$, we obtain by construction

$$E_1 = F_{k+1} - F(x_{k+1}) \leq \varepsilon_{k+1} \leq \frac{c}{\kappa}\|g_{k+1}\|^2 \leq \frac{2c}{(1-4c)\kappa}\|\nabla F(x_{k+1})\|^2, \tag{29}$$

where the last inequality comes from Corollary 1. Moreover,

$$\begin{aligned}
\|\nabla F(x_{k+1})\| &\leq \|\nabla F(x_{k+1}) - \nabla F(x_k)\| + \|\nabla F(x_k) - g_k\| + \|g_k\| \\
&\leq \kappa\|x_{k+1} - x_k\| + \sqrt{2\kappa\varepsilon_k} + \|g_k\| \\
&\leq (2 + \alpha + \sqrt{2c})\|g_k\|.
\end{aligned} \tag{30}$$

Thus,

$$E_1 \leq \frac{2c(2+\alpha+\sqrt{2c})^2}{1-4c}\frac{1}{\kappa}\|g_k\|^2. \tag{31}$$

Second, by the κ-smoothness of F, we have

$$\begin{aligned}
E_2 &= F(x_{k+1}) - F(z_k) \\
&\leq \langle \nabla F(z_k), x_{k+1} - z_k\rangle + \frac{\kappa}{2}\|x_{k+1} - z_k\|^2 \\
&\leq \|\nabla F(z_k)\|\|x_{k+1} - z_k\| + \frac{\kappa}{2}\|x_{k+1} - z_k\|^2.
\end{aligned}$$

Since $x_{k+1} - z_k = (\frac{1}{\kappa} - A_k)g_k$, we have $\|x_{k+1} - z_k\| \leq \frac{\alpha}{\kappa}\|g_k\|$. Furthermore, similarly to (30), we can bound

$$\|\nabla F(z_k)\| \leq (2 + \sqrt{2c})\|g_k\|.$$

Therefore,

$$E_2 \leq (2 + \alpha + \sqrt{2c})\frac{\alpha}{\kappa}\|g_k\|^2. \tag{32}$$

Combining (31) and (32) yields

$$E_1 + E_2 \leq \left[\frac{2c(2+\alpha+\sqrt{2c})^2}{1-4c} + (2+\alpha+\sqrt{2c})\alpha\right]\frac{1}{\kappa}\|g_k\|^2. \tag{33}$$

When $c \leq \frac{1}{72}$ and $\alpha \leq \frac{1}{32}$, we have $E_1 + E_2 \leq \frac{1}{4\kappa}\|g_k\|^2$. This together with (28) completes the proof. □

In practice, the unit stepsize is very often sufficient for the descent condition to hold, as empirically studied in Appendix C.2. The following result shows that under a specific assumption on the Moreau-Yosida envelope F, indeed the unit stepsize is always selected when the iterate are close to the optimum. The condition, called Dennis-Moré criterion [17], is classical in the literature of Quasi-Newton methods, even though we cannot formally show that it holds for the Moreau-Yosida envelope F. Indeed, the criterion requires F to be twice continuously differentiable, which is not true in general, see [32]. Therefore, the lemma below should not be seen as an formal explanation for the choice of step size $\eta_k = 1$, which we often observe in practice, but simply as a reasonable condition that leads to this choice.

Lemma 6 (**A sufficient condition for unit stepsize).** *Assume that F is twice-continuously differentiable and $\nabla^2 F$ is Lipschitz. If H_k is bounded and the Dennis-Moré criterion [17] is satisfied, i.e.*

$$\frac{\|(B_k^{-1} - \nabla^2 F(x^*)^{-1})g_k\|}{\|g_k\|} \to 0, \quad \textit{where} \quad B_k = H_k^{-1}, \tag{DM}$$

then, the descent condition (12) is satisfied with $\eta_k = 1$ when k is large enough.

We remark that the Dennis-Moré criterion we use here is slightly different from the standard one since the criterion is based on approximate gradients g_k. The proof is close to that of similar lemmas appearing in the proximal quasi-Newton literature [31], and is relegated to the appendix. Interestingly, this proof also suggests that a stronger stopping criterion ε_k such that $\varepsilon_k = o(\|g_k\|^2)$ could lead to superlinear convergence. However, such a choice of ε_k would significantly increase the complexity for solving the sub-problems, and overall degrade the global complexity.

4.4 Complexity analysis of the inner loop

In this section, we evaluate the complexity of solving the sub-problems (10) up to the desired accuracy using a linearly convergent method $\mathcal{M}$. Our main result is that all sub-problems can be solved in a constant number $T_{\mathcal{M}}$ of iterations (in expectation if the method is non-deterministic) using a warm start strategy.

Let us consider the sub-problem with an arbitrary prox center x,

$$\min_{w \in \mathbb{R}^d} \left\{ h(w) = f(w) + \frac{\kappa}{2}\|w - x\|^2 \right\}. \tag{34}$$

The number of iterations needed is determined by the ratio between the initial gap $h(w_0) - h^*$ and the desired accuracy. We are going to bound this ratio by a constant factor.

Lemma 7 (**Warm start for primal methods - smooth case).** *If f is differentiable with L-Lipschitz continuous gradients, we initialize the method $\mathcal{M}$ with $w_0 = x$. Then, we have the guarantee that*

$$h(w_0) - h^* \leq \frac{L + \kappa}{2\kappa^2}\|\nabla F(x)\|^2. \tag{35}$$

Proof. Denote by w^* the minimizer of h. Then, we have the optimality condition $\nabla f(w^*) + \kappa(w^* - x) = 0$. As a result,

$$\begin{aligned}
h(w_0) - h^* &= f(x) - \left(f(w^*) + \frac{\kappa}{2}\|w^* - x\|^2 \right) \\
&\leq f(w^*) + \langle \nabla f(w^*), x - w^* \rangle + \frac{L}{2}\|x - w^*\|^2 - \left(f(w^*) + \frac{\kappa}{2}\|w^* - x\|^2 \right) \\
&= \frac{L + \kappa}{2}\|w^* - x\|^2 \\
&= \frac{L + \kappa}{2\kappa^2}\|\nabla F(x)\|^2.
\end{aligned}$$

□

The inequality in the proof of Lemma 7 relies on the smoothness of f, which does not hold for composite problems. The next lemma addresses this issue.

Lemma 8 (**Warm start for primal methods - composite case).** *Consider the composite optimization problem $f = f_0 + \psi$, where f_0 is L-smooth. By initializing the method $\mathcal{M}$ with*

$$w_0 = \arg\min_{w \in \mathbb{R}^d} \left\{ f_0(x) + \langle \nabla f_0(x), w - x \rangle + \frac{L+\kappa}{2}\|w - x\|^2 + \psi(w) \right\}, \tag{36}$$

we have,

$$h(w_0) - h^* \leq \frac{L+\kappa}{2\kappa^2}\|\nabla F(x)\|^2.$$

Proof. We use the inequality corresponding to Lemma 2.3 in [2]: for any w,

$$h(w) - h(w_0) \geq \frac{L'}{2}\|w_0 - x\|^2 + L'\langle w_0 - x, x - w \rangle, \tag{37}$$

with $L' = L + \kappa$. Then, we apply this inequality to $w = w^*$, and

$$h(w_0) - h^* \leq -\frac{L'}{2}\|w_0 - x\|^2 - L'\langle w_0 - x, x - w^* \rangle \leq \frac{L'}{2}\|x - w^*\|^2 = \frac{L+\kappa}{2\kappa^2}\|\nabla F(x)\|^2.$$

□

We get an initialization of the same quality in the composite case as in the smooth case, by performing an additional proximal step. It is important to remark that the above analysis do not require strong convexity of f, which allows us to derive the desired inner-loop complexity.

Proposition 4 (**Inner-loop complexity for Algorithm 1).** *Consider Algorithm 1 with the warm start strategy described in Lemma 7 or in Lemma 8. Assume that the optimization method $\mathcal{M}$ applied in the inner loop produces a sequence $(w_t)_{t \geq 0}$ for each sub-problem (34) such that*

$$h(w_t) - h^* \leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^t (h(w_0) - h^*) \quad \text{for some constants} \quad C_{\mathcal{M}}, \tau_{\mathcal{M}} > 0. \tag{38}$$

Then, the stopping criterium $\varepsilon \leq \frac{1}{72\kappa}\|g\|^2$ is achieved in at most $T_{\mathcal{M}}$ iterations with

$$T_{\mathcal{M}} = \frac{1}{\tau_{\mathcal{M}}} \log\left(74 C_{\mathcal{M}} \frac{L+\kappa}{\kappa} \right).$$

Proof. Consider at iteration k, we apply $\mathcal{M}$ to approximate the proximal mapping according to x. With the given $T_{\mathcal{M}}$ (which we abbreviate by T), we have

$$\begin{aligned}
h(w_T) - h^* &\leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^T (h(w_0) - h^*) \\
&\leq C_{\mathcal{M}} e^{-\tau_{\mathcal{M}} T} (h(w_0) - h^*) \\
&\leq C_{\mathcal{M}} e^{-\tau_{\mathcal{M}} T} \frac{L+\kappa}{2\kappa^2}\|\nabla F(x)\|^2 \quad \text{(By Lemma 7 and Lemma 8)} \\
&= \frac{1}{148\kappa}\|\nabla F(x)\|^2 \\
&\leq \frac{1}{72\kappa}\|g\|^2,
\end{aligned}$$

where the last inequality follows from Lemma 2. □

Next, we extend the previous result obtained with deterministic methods $\mathcal{M}$ to randomized ones, where linear convergence is only achieved in expectation. The proof is a simple application of Lemma C.1 in [33] (see also [12] for related results on the expected complexity of randomized algorithms).

Remark 1 (When $\mathcal{M}$ is non-deterministic). *Assume that the optimization method $\mathcal{M}$ applied to each sub-problem (34) produces a sequence $(w_t)_{t\geq 0}$ such that*

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

We define the stopping time $T_{\mathcal{M}}$ by

$$T_{\mathcal{M}} = \inf\left\{t \geq 1 \mid h(w_t) - h^* \leq \frac{1}{72\kappa}\|g_t\|^2\right\}, \quad \text{where} \quad g_t = \kappa(x - w_t), \tag{39}$$

which is the random variable corresponding to the minimum number of iterations to guarantee the stopping condition (11). Then, when the warm start strategy described in Lemma 7 or in Lemma 8 is applied, the expected number of iterations satisfies

$$\mathbb{E}[T_{\mathcal{M}}] \leq \frac{1}{\tau_{\mathcal{M}}}\log\left(148 C_{\mathcal{M}}\frac{L+\kappa}{\tau_{\mathcal{M}}\kappa}\right) + 1. \tag{40}$$

Remark 2 (Checking the stopping criterium). *It is worth to notice that the stopping criterium (11), i.e. $h(w) - h^* \leq \frac{\kappa}{72}\|w - x\|^2$, can not be directly checked since h^* is unknown. Nevertheless, an upper bound on the optimality gap $h(w) - h^*$ is usually available. In particular,*

- *When f is smooth, which implies h is smooth, we have*

$$h(w) - h^* \leq \frac{1}{2(\mu+\kappa)}\|\nabla h(w)\|^2. \tag{41}$$

- *Otherwise, we can evaluate the Fenchel conjugate function, which is a natural lower bound of h^*, see Section D.2.3 in [37].*

4.5 Global complexity of QNing

Finally, we can use the previous results to upper-bound the complexity of the QNing algorithm in terms of iterations of the method $\mathcal{M}$ for minimizing f up to ε.

Proposition 5 (Worst-case global complexity for Algorithm 1). *Given a linearly-convergent method $\mathcal{M}$ satisfying (13), we apply $\mathcal{M}$ to solve the sub-problems of Algorithm 1 with the warm start strategy given in Lemma 7 or Lemma 8 up to accuracy $\varepsilon_k \leq \frac{1}{72\kappa}\|g_k\|^2$. Then, the number of iterations of the method $\mathcal{M}$ to guarantee the optimality condition $f(z_k) - f^* \leq \varepsilon$ is*

- *for μ-strongly-convex problems:*

$$O\left(T_{\mathcal{M}} \times \frac{\mu+\kappa}{\mu}\log\left(\frac{f(x_0)-f^*}{\varepsilon}\right)\right) = O\left(\frac{\mu+\kappa}{\tau_{\mathcal{M}}\mu}\log\left(\frac{f(x_0)-f^*}{\varepsilon}\right)\log\left(74 C_{\mathcal{M}}\frac{L+\kappa}{\kappa}\right)\right).$$

- *for convex problems with bounded level sets:*

$$O\left(T_{\mathcal{M}} \times \frac{2\kappa R^2}{\varepsilon}\right) = O\left(\frac{2\kappa R^2}{\tau_{\mathcal{M}}\varepsilon}\log\left(74 C_{\mathcal{M}}\frac{L+\kappa}{\kappa}\right)\right).$$

Proof. The total number of calls of method $\mathcal{M}$ is simply $T_{\mathcal{M}}$ times the number of outer-loop iterations times the potential number of line search steps at each iteration (which is hidden in the $O(.)$ notation since this number can be made arbitrarily small). □

Remark 3. *For non-deterministic methods, applying (40) yields a global complexity in expectation similar to the previous result with additional constant $2/\tau_{\mathcal{M}}$ in the last* log *factor.*

As we shall see, the global complexity of our algorithm is mainly controlled by the smoothing parameter κ. Unfortunately, under the current analysis, our algorithm QNing does not lead to an improved convergence rate in terms of the worst-case complexity bounds. It is worthwhile to underline, though, that this result is not surprising since it is often the case for L-BFGS-type methods, for which an important gap remains between theory and practice. Indeed, L-BFGS often outperforms the vanilla gradient descent method in many practical cases, but never in theory, which turns out to be the bottleneck in our analysis.

We give below the worst-case global complexity of QNing when applied to two optimization methods $\mathcal{M}$ of interest. Proposition 5 and its application to the two examples show that, in terms of worse-case complexity, the QNing scheme leaves the convergence rate almost unchanged.

Example 1. *Consider gradient descent with fixed constant step-size $1/L$ as the optimization method $\mathcal{M}$. Gradient descent (GD) minimizes f to ε accuracy in*

$$O(L/\mu \log(1/\varepsilon))$$

iterations. The complexity to achieve the same result with QNing-GD is in the worst case

$$\tilde{O}((L+\kappa)/\mu \log(1/\varepsilon)).$$

Example 2. *Consider the stochastic variance-reduced gradient (SVRG) as the optimization method $\mathcal{M}$. SVRG minimizes f to ε accuracy in*

$$O\left(\max\left\{n, \frac{L}{\mu}\right\} \log\left(\frac{1}{\varepsilon}\right)\right)$$

iterations in expectation. QNing-SVRG achieves the same result with the worst-case expected complexity

$$\tilde{O}\left(\max\left\{\frac{\mu+\kappa}{\mu}n, \frac{L+\kappa}{\mu}\right\} \log\left(\frac{n}{\varepsilon}\right)\right).$$

Choice of κ. The worst-case complexity theory does not seem to offer any guidance regarding the choice of κ for the QNing acceleration to be most effective. We heuristically assume that L-BFGS enjoys a similar performance as Nesterov's accelerated gradient method, which is often the case in practice. Then, we shall experiment with a heuristic, consisting in choosing the κ as in the related Catalyst acceleration scheme [33]. We present empirical evidence in support of this heuristic in Section 5.

5 Experiments and practical details

In this section, we present the experimental results obtained by applying QNingto several first-order optimization algorithms. We start the section by presenting the various benchmarks and practical parameter-tuning choices. Then we study the performance of QNing applied to SVRG (Section 5.3) and to the proximal gradient descent algorithm ISTA (Section 5.4), which reduces to gradient descent (GD) in the smooth case. We demonstrate that the proposed QNing can be viewed as an acceleration scheme. By applying QNing to an optimization algorithm $\mathcal{M}$, we achieve better performance then when applying $\mathcal{M}$ directly to the problem. Besides, we also compare QNing to existing stochastic variants of L-BFGS algorithm in Section 5.3. Finally, we study the behavior of QNing under different choice of parameters in Section 5.5. The code used for all the experiments is available at `https://github.com/hongzhoulin89/Catalyst-QNing/`.

5.1 Formulations and datasets

We consider three common optimization problems in machine learning and signal processing, which admit a particular structure: large finite sum, composite, strong convexity. For each formulation, we also consider a training set $(a_i, b_i)_{i=1}^n$ of n data points, where the b_i's are scalars in $\{-1, +1\}$ and the a_i's are feature vectors in $\mathbb{R}^d$. Then, the goal is to fit a linear model x in $\mathbb{R}^d$ such that the scalar b_i can be well predicted by the inner-product $\approx a_i^\top x$, or by its sign. Specifically, the three formulations we consider are listed below.

- **ℓ_2^2-regularized Logistic Regression**:

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

  which leads to a μ-strongly convex smooth optimization problem.

- **ℓ_1-regularized Linear Regression (LASSO)**:

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

  which is non smooth and convex but not strongly convex.

- **$\ell_1 - \ell_2^2$-regularized Linear Regression (Elastic-Net)**:

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

  which is based on the Elastic-Net regularization [63] leading to strongly-convex optimization problems.

Since we normalize the feature vectors a_i, a natural upper-bound on the Lipschitz constant L of the unregularized objective can be easily obtained with $L_{\text{logistic}} = 1/4$, $L_{\text{elastic-net}} = 1$ and $L_{\text{lasso}} = 1$. In the experiments, we consider relatively ill-conditioned problems with the regularization parameter $\mu = 1/(100n)$. The ℓ_1-regularization parameter is set to $\lambda = 1/n$ for the Elastic-Net formulation; for the Lasso problem, we consider a logarithmic grid $10^i/n$, with $i = -3, -2, \ldots, 3$, and we select the parameter λ that provides a sparse optimal solution closest to 10% non-zero coefficients.

Datasets. We consider five standard machine learning datasets with different characteristics in terms of size and dimension, which are described below:

name	covtype	alpha	real-sim	MNIST-CKN	CIFAR-CKN
n	581 012	250 000	72 309	60 000	50 000
d	54	500	20 958	2 304	9 216

The first three data sets are standard machine learning data sets from LIBSVM [13]. We normalize the features, which provides a natural estimate of the Lipschitz constant as mentioned previously. The last two data sets are coming from computer vision applications. MNIST and CIFAR-10 are two image classification data sets involving 10 classes. The feature representation of each image was computed using an unsupervised convolutional kernel network [39]. We focus here on the task of classifying class #1 vs. other classes.

5.2 Choice of hyper-parameters and variants

We now discuss the choice of default parameters used in the experiments as well as the different variants. First, to deal with the high-dimensional nature of the data, we systematically use the L-BFGS metric H_k and maintain the positive definiteness by skipping updates when necessary (see [20]).

Choice of method $\mathcal{M}$. We apply QNing to proximal SVRG algorithm [60] and proximal gradient algorithm. The proximal SVRG algorithm is an incremental algorithm that is able to exploit the finite-sum structure of the objective and can deal with the composite regularization. We also consider the gradient descent algorithm and its proximal variant ISTA, which allows us to perform a comparison with the natural baselines FISTA [2] and L-BFGS.

Stopping criterion for the inner loop. The default stopping criterion consists of solving each sub-problem with accuracy ε_k such that $\varepsilon_k \leq \frac{1}{72}\|g_k\|^2$. Although we have shown that such accuracy is attainable in some constant $T = \tilde{O}(n)$ number of iterations for SVRG with the choice $\kappa = L/2n$, a natural heuristic also proposed in Catalyst [34] consists of performing exactly one pass over the data $T = n$ in the inner loop without checking any stopping criterion. In particular, for gradient descent or ISTA, one pass over the data means a single gradient step, because the evaluation of the full gradient requires passing through the entire dataset. When applying QNing to SVRG and ISTA, we call the default algorithm using stopping criterion (23) **QNing-SVRG**, **QNing-ISTA** and the one-pass variant **QNing-SVRG1**, **QNing-ISTA1**, respectively.

Choice of regularization parameter κ. We choose κ as in the Catalyst algorithm [34], which is $\kappa = L$ for gradient descent/ISTA and $\kappa = L/2n$ for SVRG. Indeed, convergence of L-BFGS is hard to characterize and its theoretical rate of convergence can be pessimistic as shown in our theoretical analysis. Noting that for smooth functions, L-BFGS often outperforms Nesterov's accelerated gradient method, it is reasonable to expect QNing achieves a similar complexity bound as Catalyst. Later in Section 5.5, we make a comparison between different values of κ to demonstrate the effectiveness of this strategy.

Choice of limited memory parameter l. The default setting is $l = 100$. We show later in Section 5.5 a comparison with different values to study the influence of this parameter.

Implementation of the line search. As mentioned earlier, we consider the stepsizes η_k in the set $\{1, 1/2, 1/4, 1/8, 0\}$ and select the largest one that satisfies the descent condition.

Evaluation metric. For all experiments, we use the number of gradient evaluations as a measure of complexity, assuming this is the computational bottleneck of all methods considered. This is indeed the case here since the L-BFGS step cost $O(dl)$ floating-point operations [50], whereas evaluating the gradient of the full objective costs $O(nd)$, with $l \ll n$.

5.3 QNing-SVRG for minimizing large sums of functions

We now apply QNing to SVRG and compare different variants.

- **SVRG**: the Prox-SVRG algorithm of [60] with default parameters $m = 1$ and $\eta = 1/L$, where L is the upper-bound on Lipschitz constant of the gradient, as described in the Section 5.1.
- **Catalyst-SVRG**: The Catalyst meta-algorithm of [34] applied to Prox-SVRG, using the variant (C3) that performs best among the different variants of Catalyst.
- **L-BFGS/Orthant**: Since implementing effectively L-BFGS with a line-search algorithm is a bit involved, we use the implementation of Mark Schmidt[2], which has been widely used in other comparisons [56]. In particular, the Orthant-wise method follows the algorithm developed in [1]. We use L-BFGS for the logistic regression experiment and the Orthant-wise method [1] for elastic-net and lasso experiments. The limited memory parameter l is set to 100.
- **QNing-SVRG**: the algorithm according to the theory by solving the sub-problems until $\varepsilon_k \leq \frac{1}{72}\|g_k\|^2$.
- **QNing-SVRG1**: the one-pass heuristic.

The result of the comparison is presented in Figure 1 and leads to the conclusions below, showing that **QNing-SVRG1** is a safe heuristic, which never decreases the speed of the method **SVRG**:

- **L-BFGS/Orthant** is less competitive than other approaches that exploit the sum structure of the objective, except on the dataset real-sim; the difference in performance with the SVRG-based approaches can be very important (see dataset alpha).

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

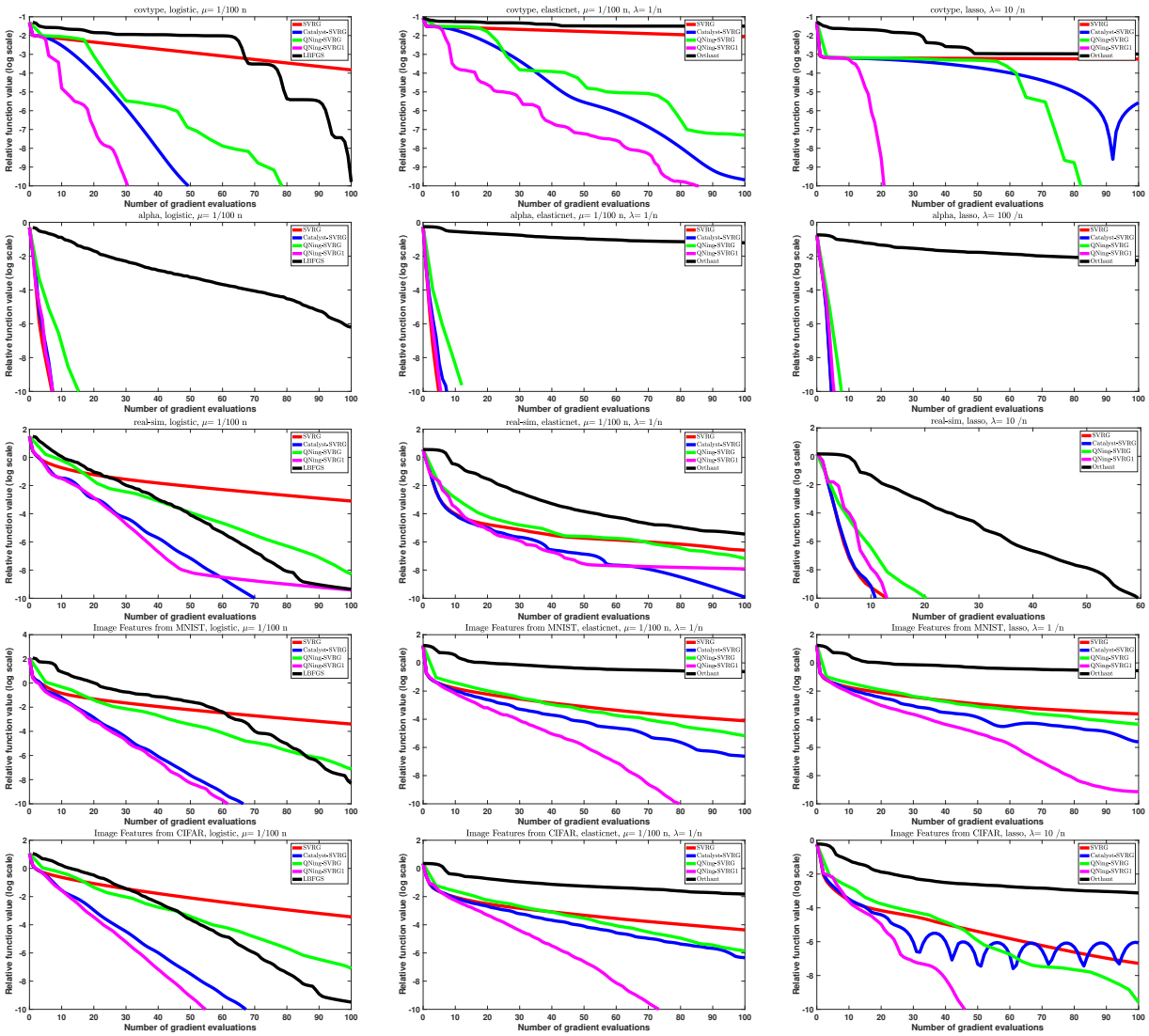

Figure 1: Experimental study of the performance of QNing-SVRG for minimizing large sums of functions. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

- **QNing-SVRG1** is significantly faster than or on par with **SVRG** and **QNing-SVRG**.
- **QNing-SVRG** is significantly faster than, or on par with, or only slightly slower than **SVRG**.
- **QNing-SVRG1** is significantly faster, or on par with **Catalyst-SVRG**. This justifies our choice of κ which assumes "a priori" that L-BFGS performs as well as Nesterov's method.

So far, we have shown that applying QNing with SVRG provides a significant speedup compared to the original SVRG algorithm or other acceleration scheme such as Catalyst. Now we compare our algorithm to other variable metric approaches including Proximal L-BFGS [31] and Stochastic L-BFGS [44]:

- **Proximal L-BFGS:** We apply the Matlab package PNOPT[3] implemented by [31]. The sub-problems are solved by the default algorithm up to desired accuracy. We consider one sub-problem as one gradient evaluation in our plot, even though it often requires multiple passes.
- **Stochastic L-BFGS** (for smooth objectives): We apply the Matlab package StochBFGS[4] implemented by [44]. We consider the 'prev' variant which has the best practical performance.

The result of the comparison is presented in Figure 2 and we observe that **QNing-SVRG1** is significantly faster than Proximal L-BFGS and Stochastic L-BFGS:

- **Proximal L-BFGS** often outperforms **Orthant**-based methods but it is less competitive than QNing.
- **Stochastic L-BFGS** is very sensitive to parameters and data since the variable metric is based on stochastic information which may have high variance. It performs very well on dataset covtype but becomes less competitive on other datasets. Moreover, it only applies to smooth problems.

The previous results are complemented by Appendix C.1, which also presents some comparison in terms of outer-loop iterations, regardless of the cost of the inner-loop.

5.4 QNing-ISTA and comparison with L-BFGS

The previous experiments have included a comparison between L-BFGS and approaches that are able to exploit the sum structure of the objective. It is then interesting to study the behavior of QNing when applied to a basic proximal gradient descent algorithm such as ISTA. Specifically, we now consider

- **GD/ISTA**: the classical proximal gradient descent algorithm ISTA [2] with back-tracking line-search to automatically adjust the Lipschitz constant of the gradient objective;
- **Acc-GD/FISTA**: the accelerated variant of ISTA from [2].
- **QNing-ISTA**, and **QNing-ISTA1**, as in the previous section replacing SVRG by GD/ISTA.

The results are reported in Figure 3 and lead to the following conclusions

- **L-BFGS** is slightly better on average than **QNing-ISTA1** for smooth problems, which is not surprising since we use a state-of-the-art implementation with a well-calibrated line search.
- **QNing-ISTA1** is always significantly faster than **ISTA** and **QNing-ISTA**.
- The **QNing-ISTA** approaches are significantly faster than **FISTA** in 12 cases out of 15.
- There is no clear conclusion regarding the performance of the **Orthant-wise** method vs other approaches. For three datasets, covtype, alpha and mnist, **QNing-ISTA** is significantly better than **Orthant-wise**. However, on the other two datasets, the behavior is different **Orthant-wise** method outperforms **QNing-ISTA**.

[3]available here `https://web.stanford.edu/group/SOL/software/pnopt`
[4]available here `https://perso.telecom-paristech.fr/rgower/software.html`

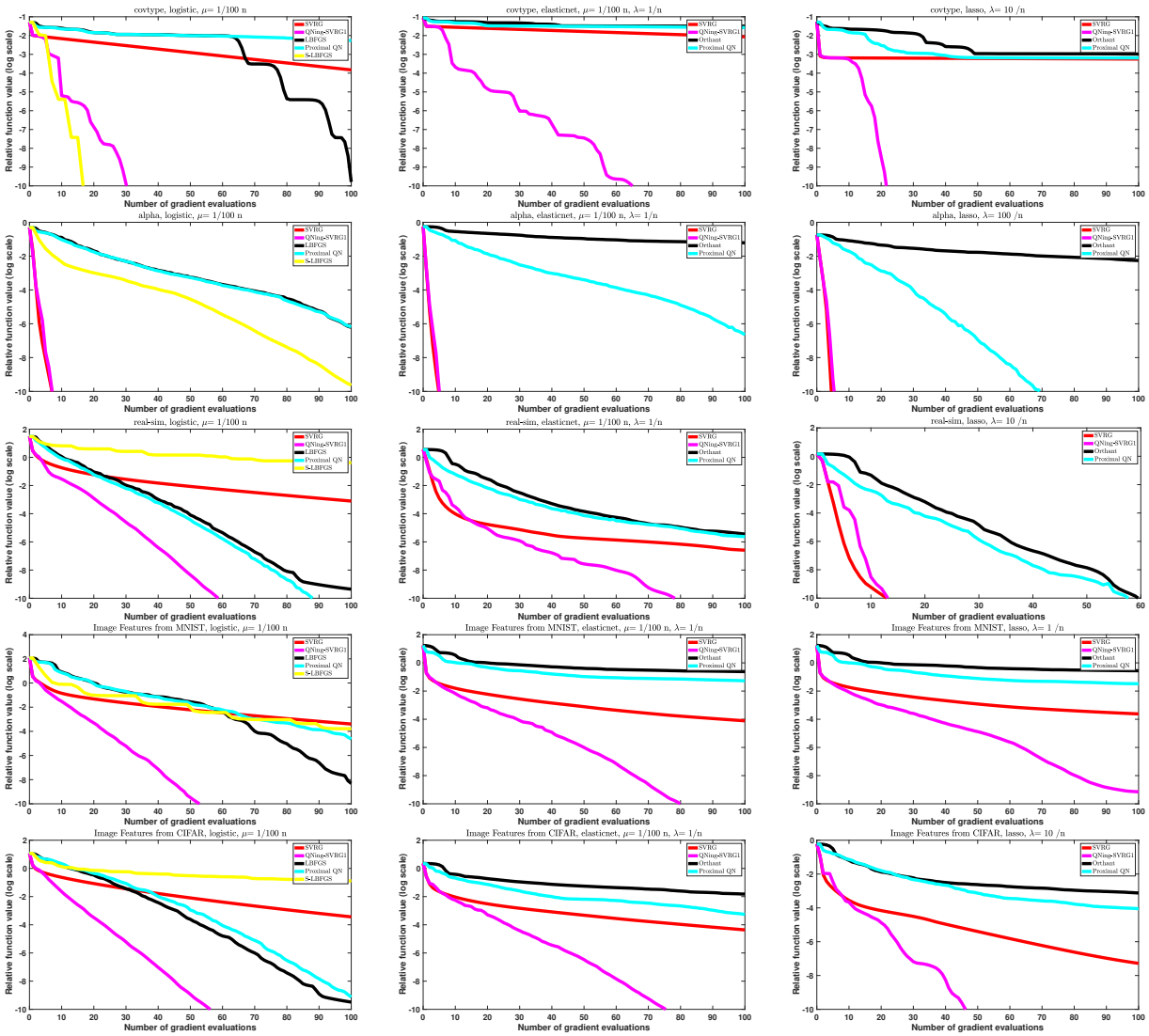

Figure 2: Comparison to proximal L-BFGS and Stochastic L-BFGS. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

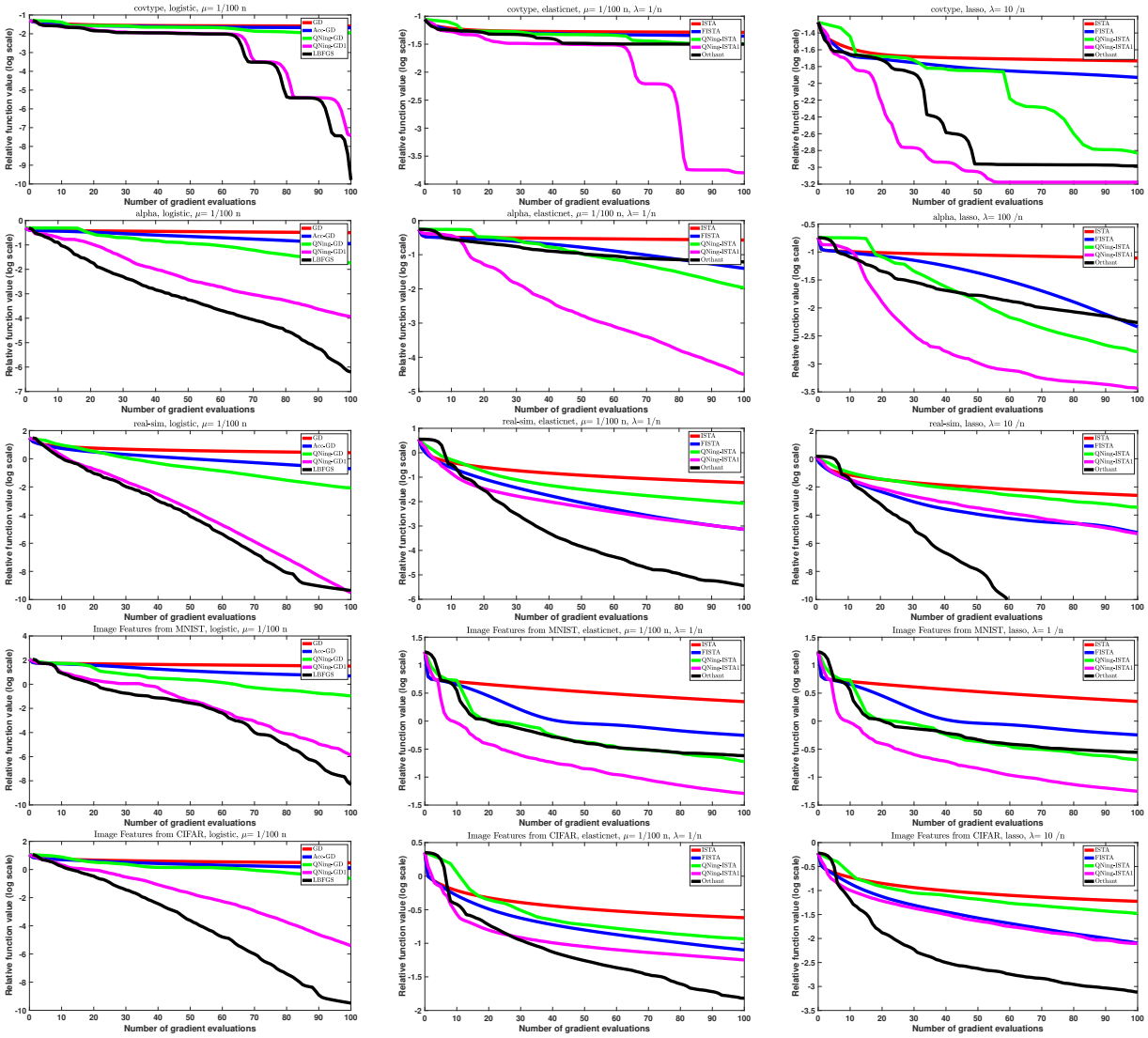

Figure 3: Experimental study of the performance of QNing-ISTA. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

5.5 Experimental study of hyper-parameters l and κ

In this section, we study the influence of the limited memory parameter l and of the regularization parameter κ in QNing. More precisely, we start with the parameter l and try the method **QNing-SVRG1** with the values $l = 1, 2, 5, 10, 20, 100$. Note that all previous experiments were conducted with $l = 100$, which is the most expensive in terms of memory and computational cost for the L-BFGS step. The results are presented in Figure 4. Interestingly, the experiment suggests that having a large value for l is not necessarily the best choice, especially for composite problems where the solution is sparse, where $l = 10$ seems to be a reasonable choice in practice.

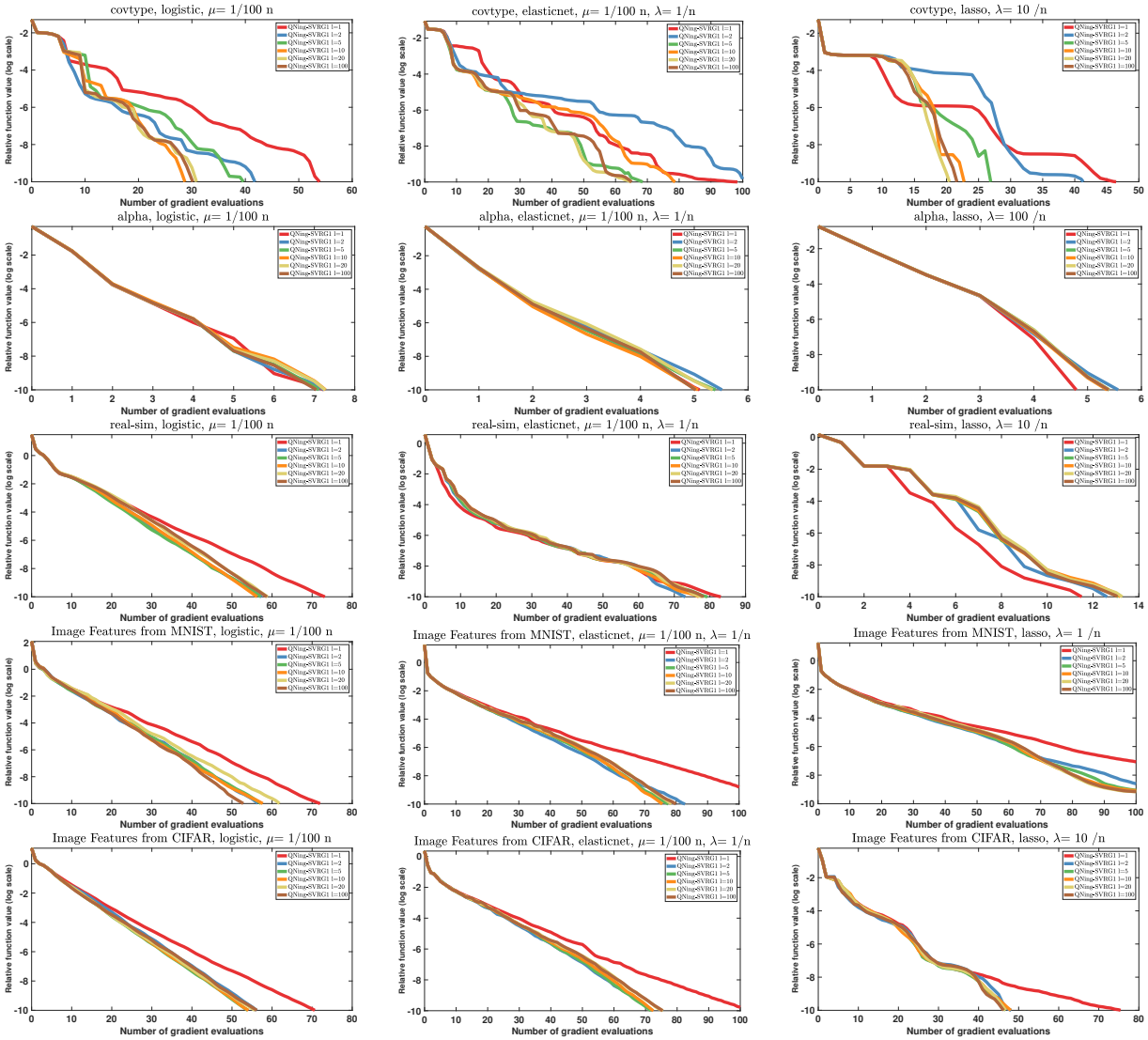

Figure 4: Experimental study of influence of the limited-memory parameter l for QNing-SVRG1. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

The next experiment consists of studying the robustness of QNing to the smoothing parameter κ. We present in Figure 5 an experiment by trying the values $\kappa = 10^i \kappa_0$, for $i = -3, -2, \ldots, 2, 3$, where κ_0 is the default parameter that we used in the previous experiments. The conclusion is clear: QNing clearly slows

down when using a larger smoothing parameter than κ_0, but it is very robust to small values of κ (and in fact it even performs better for smaller values than κ_0).

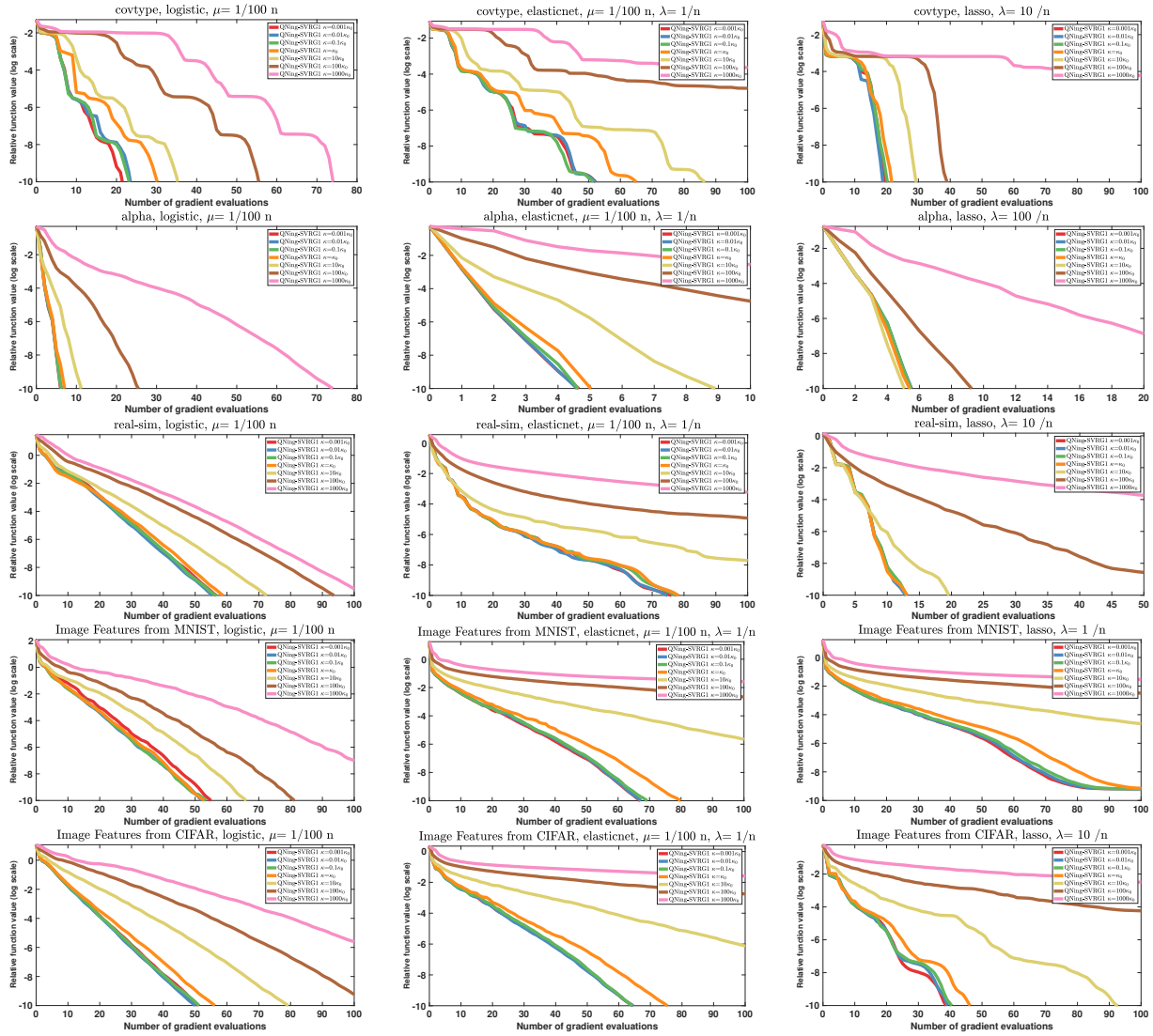

Figure 5: Experimental study of influence of the smoothing parameter κ for QNing-SVRG1. κ_0 denotes the default choice used in the previous experiments. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

6 Discussions and concluding remarks

A few questions naturally arise regarding the QNing scheme: one may wonder whether or not our convergence rates may be improved, or if 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

In this paper, we have established the linear convergence of QNing for strongly convex objectives when sub-problems are solved with enough accuracy. Since QNing uses Quasi-Newton steps, one might have expected a superlinear convergence rate as several Quasi-Newton algorithms often enjoy [11]. The situation is as follows. Consider the BFGS Quasi-Newton algorithm (without limited memory). As shown in [14], if the sequence $(\varepsilon_k)_{k\geq 0}$ decreases super-linearly, then, it is possible to design a scheme similar to QNing that indeed enjoys a super-linear convergence rate. There are two major downsides though. The scheme of [14] with such a 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 QNing algorithm. Indeed, the convergence rate of QNing 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 2 (ApproxGradient) 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 [3], whose family also encompasses the so-called Nesterov smoothing [3]. Other ways to smooth a function include randomization techniques [18] or 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 [3], this property is also shared by other types of infimal convolutions. Therefore, QNing 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, QNing, to accelerate existing first-order optimization algorithms with quasi-Newton-type rules to update a variable metric along the iterations. QNing'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 questions in the literature about L-BFGS methods.

Acknowledgements

This work was supported by the ERC grant SOLARIS (number 714381), a grant from ANR (MACARON project ANR-14-CE23-0003-01), and the program "Learning in Machines and Brains" (CIFAR).

References

[1] G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. In *International Conferences on Machine Learning (ICML)*, 2007.

[2] 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.

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

[4] S. Becker and J. Fadili. A quasi-newton proximal splitting method. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2618–2626, 2012.

[5] D. P. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, 169:337–375, 2018.

[13] C. Chang and C. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

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

[15] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2014.

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

[17] J. E. Dennis and J. J. Moré. A characterization of superlinear convergence and its application to quasi-newton methods. *Mathematics of computation*, 28(126):549–560, 1974.

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

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

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

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

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

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

[24] S. Ghadimi, G. Lan, and H. Zhang. Generalized Uniformly Optimal Methods for Nonlinear Programming. *arxiv:1508.07384*, 2015.

[25] H. Ghanbari and K. Scheinberg. Proximal quasi-newton methods for convex optimization. *arXiv:1607.03081*, 2016.

[26] R. M. Gower, D. Goldfarb, and P. Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. In *International Conferences on Machine Learning (ICML)*, 2016.

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

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

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

[30] C. Lee and S. J. Wright. Inexact successive quadratic approximation for regularized optimization. *arXiv:1803.01298*, 2018.

[31] J. Lee, Y. Sun, and M. Saunders. Proximal Newton-type methods for convex optimization. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2012.

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

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

[34] H. Lin, J. Mairal, and Z. Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. *Journal of Machine Learning Research (JMLR)*, 18(212):1–54, 2018.

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

[36] X. Liu, C.-J. Hsieh, J. D. Lee, and Y. Sun. An inexact subsampled proximal newton-type method for large-scale machine learning. *arXiv:1708.08552*, 2017.

[37] J. Mairal. *Sparse coding for machine learning, image processing and computer vision.* PhD thesis, Cachan, Ecole normale supérieure, 2010.

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

[39] J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2016.

[40] J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

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

[42] A. Mokhtari and A. Ribeiro. Global convergence of online limited memory bfgs. *Journal of Machine Learning Research (JMLR)*, 16(1):3151–3181, 2015.

[43] J.-J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *CR Acad. Sci. Paris Sér. A Math*, 255:2897–2899, 1962.

[44] P. Moritz, R. Nishihara, and M. I. Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[45] 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.

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

[47] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

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

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

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

[51] 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.

[52] 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.

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

[54] K. Scheinberg and X. Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. *Mathematical Programming*, 160(1):495–529, 2016.

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

[56] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 160(1):83–112, 2017.

[57] N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-newton method for online convex optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[58] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1):105–145, 2016.

[59] L. Stella, A. Themelis, and P. Patrinos. Forward–backward quasi-newton methods for nonsmooth optimization problems. *Computational Optimization and Applications*, 67(3):443–487, 2017.

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

[61] K. Yosida. Functional analysis. *Berlin-Heidelberg*, 1980.

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

[63] 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.

A Proof of Proposition 3

First, we show that the Moreau envelope F inherits the bounded level set property from f.

Definition 2. *We say that a convex function f has bounded level sets if f attains its minimum at x^* in $\mathbb{R}^d$ and for any x, there exists $R_x > 0$ such that*

$$\forall y \in \mathbb{R}^d \quad s.t. \quad f(y) \leq f(x) \quad then \quad \|y - x^*\| \leq R_x.$$

Lemma 9. *If f has bounded level sets, then its Moreau envelope F has bounded level sets as well.*

Proof. First, from Proposition 1, the minimum of F is attained at x^*. Next, we reformulate the bounded level set property by contraposition: for any x, there exists $R_x > 0$ such that

$$\forall y \in \mathbb{R}^d \quad \text{s.t.} \quad \|y - x^*\| > R_x \quad \text{then} \quad f(y) > f(x).$$

Given x in $\mathbb{R}^d$, we show that

$$\forall y \in \mathbb{R}^d \quad \text{s.t.} \quad \|y - x^*\| > \sqrt{\frac{2(f(x) - f^*)}{\kappa}} + R_x \quad \text{then} \quad F(y) > F(x).$$

Let y satisfies the above inequality, by definition,

$$F(y) = f(p(y)) + \frac{\kappa}{2}\|p(y) - y\|^2.$$

From the triangle inequality,

$$\|y - p(y)\| + \|p(y) - x^*\| \geq \|y - x^*\| > \sqrt{\frac{2(f(x) - f^*)}{\kappa}} + R_x.$$

Then either $\|y - p(y)\| > \sqrt{\frac{2(f(x)-f^*)}{\kappa}}$ or $\|p(y) - x^*\| > R_x$.

- If $\|y - p(y)\| > \sqrt{\frac{2(f(x)-f^*)}{\kappa}}$, then

$$F(y) = f(p(y)) + \frac{\kappa}{2}\|p(y) - y\|^2 > f(p(y)) + f(x) - f^* \geq f(x) \geq F(x).$$

- If $\|p(y) - x^*\| > R_x$, then

$$F(y) = f(p(y)) + \frac{\kappa}{2}\|p(y) - y\|^2 \geq f(p(y)) > f(x) \geq F(x).$$

This completes the proof. □

We are now in shape to prove the proposition.

Proof. From (24), we have

$$F(x_{k+1}) \leq F(x_k) - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2.$$

Thus $F(x_k)$ is decreasing. From the bounded level set property of F, there exists $R > 0$ such that $\|x_k - x^*\| \leq R$ for any k. By the convexity of F, we have

$$F(x_k) - F^* \leq \langle \nabla F(x_k), x_k - x^* \rangle \leq \|\nabla F(x_k)\|x_k - x^*\|\| \leq R\|\nabla F(x_k)\|.$$

Therefore,

$$\begin{aligned} F(x_{k+1}) - F^* &\leq F(x_k) - F^* - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 \\ &\leq F(x_k) - F^* - \frac{(F(x_k) - F^*)^2}{8\kappa R^2}. \end{aligned}$$

Let us define $r_k \triangleq f(x_k) - f^*$. Thus,

$$\frac{1}{r_{k+1}} \geq \frac{1}{r_k(1 - \frac{r_k}{8\kappa R^2})} \geq \frac{1}{r_k}\left(1 + \frac{r_k}{8\kappa R^2}\right) = \frac{1}{r_k} + \frac{1}{8\kappa R^2}.$$

Then, after exploiting the telescoping sum,

$$\frac{1}{r_{k+1}} \geq \frac{1}{r_0} + \frac{k+1}{8\kappa R^2} \geq \frac{k+1}{8\kappa R^2}.$$

□

B Proof of Lemma 6

Proof. Let us denote $\delta_k = -B_k^{-1} g_k$ and show that $x_k + \delta_k$ satisfies the descent condition when x_k is close to the optimum. More precisely, we show that

$$F(x_k + \delta_k) \leq F(x_k) - \frac{3}{10\kappa}\|g_k\|^2 + o(\|g_k\|^2), \tag{42}$$

which is sufficient to conclude the proof. By the Lipschitz Hessian assumption, we have

$$\begin{aligned} F(x_k + \delta_k) - F(x_k) &\leq \nabla F(x_k)^T \delta_k + \frac{1}{2}\delta_k^T \nabla^2 F(x_k)\delta_k + \frac{L_2}{6}\|\delta_k\|^3 \\ &= (\nabla F(x_k) - g_k)^T \delta_k + g_k^T \delta_k + \frac{1}{2}\delta_k^T(\nabla^2 F(x_k) - B_k)\delta_k + \underbrace{\frac{1}{2}\delta_k^T B_k \delta_k}_{=-\frac{1}{2}g_k^T\delta_k} + \frac{L_2}{6}\|\delta_k\|^3 \\ &= \underbrace{\frac{1}{2}g_k^T\delta_k}_{E_1} + \underbrace{(\nabla F(x_k) - g_k)^T\delta_k}_{E_2} + \underbrace{\frac{1}{2}\delta_k^T(\nabla^2 F(x_k) - B_k)\delta_k}_{E_3} + \underbrace{\frac{L_2}{6}\|\delta_k\|^3}_{E_4} \end{aligned}$$

We are going upper bound each term one by one. First,

$$\begin{aligned}
E_1 &= \frac{1}{2} g_k^T \delta_k = -\frac{1}{2} g_k B_k^{-1} g_k \\
&= -\frac{1}{2} g_k \nabla^2 F(x^*)^{-1} g_k - \frac{1}{2} g_k (B_k^{-1} - \nabla^2 F(x^*)^{-1}) g_k \\
&\leq -\frac{1}{2\kappa} \|g_k\|^2 + o(\|g_k\|^2),
\end{aligned}$$

where the last inequality uses (DM) and the κ-smoothness of F which implies $\nabla^2 F(x^*) \preceq \kappa I$. Second,

$$\begin{aligned}
E_2 &= (\nabla F(x_k) - g_k)^T \delta_k \leq \|\nabla F(x_k) - g_k\| \|\delta_k\| \\
&\leq \sqrt{2c} \|g_k\| \|B_k^{-1} g_k\| \\
&\leq \sqrt{2c} \|g_k\| (\|\nabla^2 F(x^*)^{-1} g_k\| + \|B_k^{-1} - \nabla^2 F(x^*)^{-1} g_k\|) \\
&\leq \sqrt{2c} \frac{1}{\mu_F} \|g_k\|^2 + o(\|g_k\|^2) \\
&= \frac{1}{4\sqrt{2}\kappa} \|g_k\|^2 + o(\|g_k\|^2).
\end{aligned}$$

Third,

$$\begin{aligned}
E_3 &= \frac{1}{2} \delta_k^T (\nabla^2 F(x_k) - B_k) \delta_k \leq \frac{1}{2} \|\delta_k\| \|(\nabla^2 F(x_k) - B_k) \delta_k\| \\
&\leq \frac{1}{2} \|\delta_k\| \left(\|(\nabla^2 F(x_k) - \nabla^2 F(x^*)) \delta_k\| + \|(\nabla^2 F(x^*) - B_k) \delta_k\| \right) \\
&\leq \frac{L_2}{2} \|x_k - x^*\| \|\delta_k\|^2 + \|\nabla^2 F(x^*)\| \|(B_k^{-1} - \nabla^2 F(x^*)^{-1}) g_k\| \\
&= o(\|g_k\|^2),
\end{aligned}$$

where the last line comes from (DM) and the fact that $\|x_k - x^*\| \to 0$. Last, since B_k are bounded and $\|g_k\| \to 0$, we have

$$E_4 = \frac{L_2}{6} \|\delta_k\|^3 \leq \frac{L_2}{6} \|B_k^{-1}\|^3 \|g_k\|^3 = o(\|g_k\|^2).$$

Summing up above four inequalities yields (42).

□

C Additional experiments

In this section, we provide additional experimental results including experimental comparisons in terms of outer loop iterations, and an empirical study regarding the choice of the unit step size $\eta_k = 1$.

C.1 Comparisons in terms of outer-loop iterations

In the main paper, we have used the number of gradient evaluations as a natural measure of complexity. Here, we also provide a comparison in terms of outer-loop iterations, which does not take into account the complexity of solving the sub-problems. While interesting, the comparison artificially gives an advantage to the stopping criteria (11) since achieving it usually requires multiple passes.

The result of the comparison is presented in Figure 6. We observe that the theoretical grounded variant QNing-SVRG always outperform the one-pass heuristic QNing-SVRG1. This is not surprising since the sub-problems are solved more accurately in the theoretical grounded variant. However, once we take the complexity of the sub-problems into account, QNing-SVRG never outperforms QNing-SVRG1. This suggests that it is not beneficial to solve the sub-problem up to high accuracy as long as the algorithm converge.

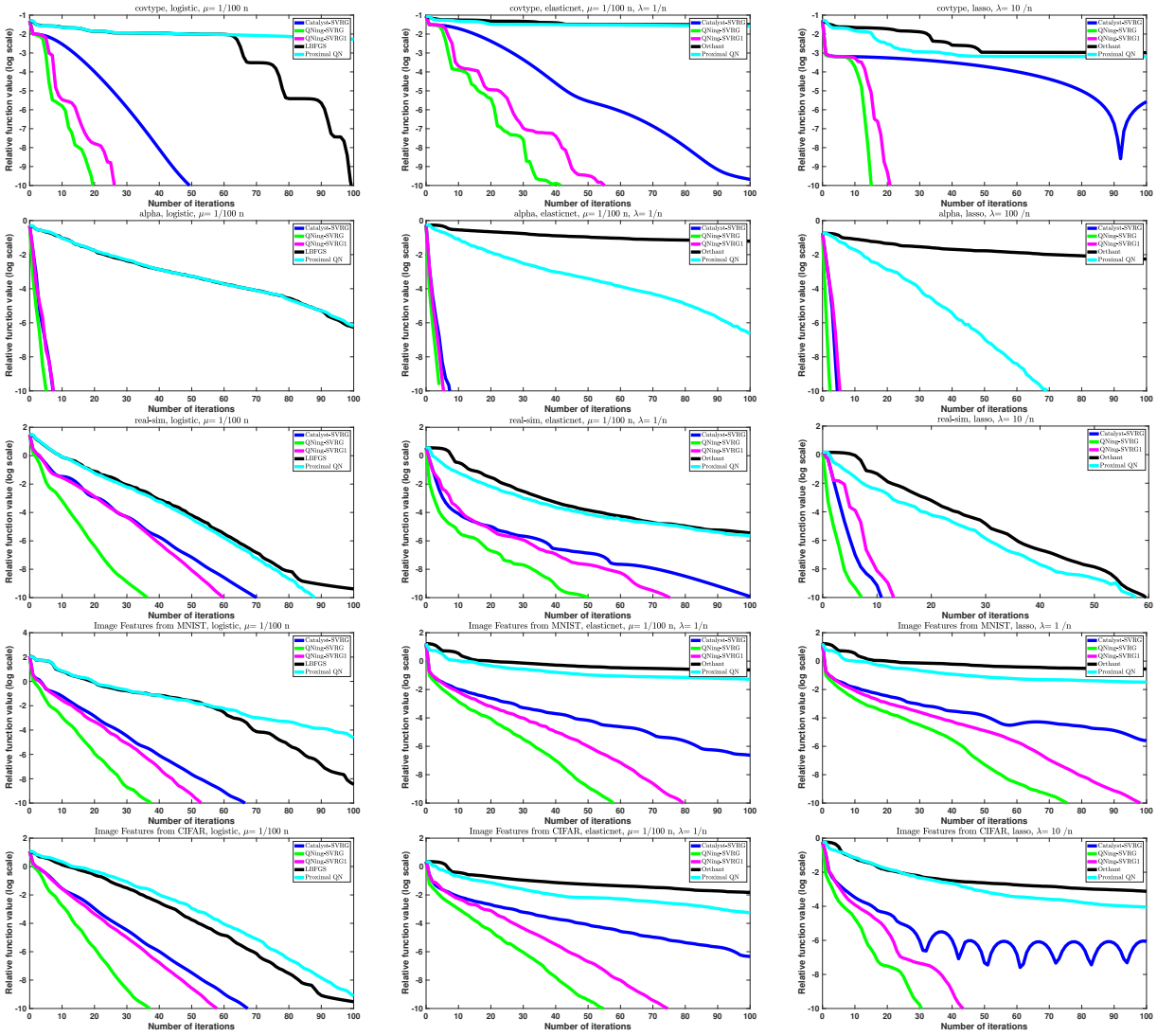

Figure 6: Experimental study of the performance of QNing-SVRG respect to the number of outer iterations.

C.2 Empirical frequency of choosing the unit stepsize

In this section, we evaluate how often the unit stepsize is taken in the line search. When the unit stepsize is taken, the variable metric step provides sufficient decrease, which is the key for acceleration. The statistics of QNing-SVRG1 (one-pass variant) and QNing-SVRG (the sub-problems are solved until the stopping criteria (11) is satisfed) are given in Table 1 and Table 2, respectively. As we can see, for most of the iterations ($> 90\%$), the unit stepsize is taken.

Table 1: Relative frequency of picking the unit stepsize $\eta_k = 1$ of QNing-SVRG1

	Logistic		**Elastic-net**		**Lasso**	
covtype	24/27	89%	54/56	96%	19/21	90%
alpha	8/8	100%	6/6	100%	6/6	100%
real-sim	60/60	100%	71/76	93%	14/14	100%
mnist	53/53	100%	80/80	100%	100/100	100%
cifar-10	58/58	100%	75/75	100%	42/44	95%

The first column is in the form N/D, where N is the number of times over the iterations the unit stepsize was picked and D is the total number of iterations. The total number of iterations D varies a lot since we stop our algorithm as soon as the relative function gap is smaller than 10^{-10} or the maximum number of iterations 100 is reached. It implicitly indicates how easy the problem is.

Table 2: Relative frequency of picking the unit stepsize $\eta_k = 1$ of QNing-SVRG

	Logistic		**Elastic-net**		**Lasso**	
covtype	18/20	90%	23/25	92%	16/16	100%
alpha	6/6	100%	4/4	100%	3/3	100%
real-sim	27/27	100%	20/23	87%	8/8	100%
mnist	27/27	100%	28/28	100%	28/28	100%
cifar-10	25/25	100%	29/29	100%	31/31	100%

The setting are the same as in Table 1.