Generic acceleration schemes for gradient-based optimization in machine learning

Hongzhou Lin

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

HONGZHOU LIN

Thèse dirigée par **Zaid HARCHAOUI**
et codirigée par **Julien MAIRAL**, chercheur, Inria
préparée au sein du **Laboratoire Jean Kuntzmann**

dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Algorithmes d'accélération générique pour les méthodes d'optimisation en apprentissage statistique

Generic acceleration schemes for gradient-based optimization in machine learning

Thèse soutenue publiquement le **16 novembre 2017**,
devant le jury composé de :

Monsieur ZAID HARCHAOUI
PROFESSEUR ASSISTANT, UNIVERSITE DE WASHINGTON - ETATS-UNIS, Directeur de thèse
Monsieur JULIEN MAIRAL
CHARGE DE RECHERCHE, INRIA DELEGATION ALPES, Co-directeur de thèse
Monsieur FRANÇOIS GLINEUR
PROFESSEUR, UNIVERSITE CATHOLIQUE LOUVAIN - BELGIQUE, Rapporteur
Monsieur GABRIEL PEYRE
DIRECTEUR DE RECHERCHE, CNRS DELEGATION PARIS, Rapporteur
Monsieur JERÔME MALICK
DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES, Président
Madame STEFANIE JEGELKA
PROFESSEUR ASSISTANT, MIT (CAMBRIDGE) - USA, Examinateur
Monsieur ALEXANDRE D'ASPREMONT
DIRECTEUR DE RECHERCHE, CNRS DELEGATION PARIS, Examinateur

Résumé

Les problèmes d'optimisation apparaissent naturellement pendant l'entrainement de modèles d'apprentissage supervisés. Un exemple typique est le problème de minimisation du risque empirique (ERM), qui vise à trouver un estimateur en minimisant le risque sur un ensemble de données. Le principal défi consiste à concevoir des algorithmes d'optimisation efficaces permettant de traiter un grand nombre de données dans des espaces de grande dimension. Dans ce cadre, les méthodes classiques d'optimisation, telles que l'algorithme de descente de gradient et sa variante accélérée, sont coûteux en termes de calcul car elles nécessitent de passer à travers toutes les données à chaque évaluation du gradient. Ce défaut motive le développement de la classe des algorithmes incrémentaux qui effectuent des mises à jour avec des gradients incrémentaux. Ces algorithmes réduisent le coût de calcul par itération, entraînant une amélioration significative du temps de calcul par rapport aux méthodes classiques. Une question naturelle se pose : serait-il possible d'accélérer davantage ces méthodes incrémentales ? Nous donnons une réponse positive en introduisant plusieurs schémas d'accélération génériques, qui s'appliquent aux méthodes d'optimisation de premier ordre.

Dans le chapitre 2, nous développons une variante proximale de l'algorithme Finito/MISO, qui est une méthode incrémentale initialement conçue pour des problèmes lisses et fortement convexes. Nous introduisons une étape proximale dans la mise à jour de l'algorithme pour prendre en compte la pénalité de régularisation qui est potentiellement non lisse. Lorsque le problème est fortement convexe, l'algorithme admet un taux de convergence linéaire similaire à l'algorithme Finito/MISO original. L'avantage de cette méthode est qu'elle donne systématiquement un certificat d'optimalité à chaque itération, basé sur des bornes inférieures quadratiques. Contrairement aux algorithmes duals comme SDCA, la construction de notre algorithme est purement primale. Ni la preuve de convergence ni l'algorithme ne nécessitent la dualité de Fenchel, ce qui le rend facile à évaluer en pratique.

Dans le chapitre 3, nous introduisons un schéma d'accélération générique pour les problèmes convexes, appelé Catalyst, lequel accélère les méthodes d'optimisation de premier ordre dans le sens de Nesterov. Il s'applique à une grande classe d'algorithmes, y compris la descente de gradient, la descente de gradient par coordonnées, des algorithmes incrémentaux tels que SAG, SAGA, SDCA, SVRG, Finito/MISO et leurs variantes proximales. Pour toutes ces méthodes, nous montrons qu'une accélération est obtenue en appliquant Catalyst sur les problèmes convexes. Il est important de remarquer que notre approche est capable de traiter des objectifs non fortement convexes même si certaines méthodes incrémentales ne sont définies que sur des problèmes fortement convexes. En outre, Catalyst peut être vu comme une variante approchée de l'algorithme de point proximal accéléré. À chaque itération, il utilise la méthode que l'on souhaite accélérer pour résoudre un sous problème,

qui revient à approcher un opérateur proximal. La clé de l'accélération est de choisir attentivement un critère d'arrêt et de contrôler l'effort de calcul requis. Nous montrons ensuite que l'accélération est bien effectuée en pratique.

Dans le chapitre 4, nous présentons une seconde approche générique qui applique les principes Quasi-Newton pour accélérer les méthodes de premier ordre, appelée QuickeNing. Le schéma s'applique à la même classe de méthodes que Catalyst. À notre connaissance, QuickeNing est le premier algorithme de type Quasi-Newton compatible avec les objectifs composites et la structure de somme finie. L'algorithme admet une simple interprétation comme une combinaison de l'algorithme L-BFGS et de la régularisation Moreau-Yosida. Nous proposons une nouvelle stratégie pour déterminer le pas à chaque itération afin d'éviter des recherches linéaires coûteuses, ce qui rend l'algorithme pratique à l'utilisation. Nous montrons expérimentalement que QuickeNing apporte une amélioration significative par rapport aux méthodes concurrentes dans les problèmes d'apprentissage à grande échelle.

Nous concluons cette thèse en proposant une extension de l'algorithme Catalyst au cas non convexe. Il s'agit d'un travail en collaboration avec Dr. Courtney Paquette et Pr. Dmitriy Drusvyatskiy, de l'Université de Washington, et mes encadrants de thèse. Le point fort de cette approche réside dans sa capacité à s'adapter automatiquement à la convexité. En effet, aucune information sur la convexité de la fonction n'est nécessaire avant de lancer l'algorithme. Ceci est réalisé par une étape spécifique, appelée auto-adaptation, qui exploite l'information de convexité locale. Lorsque l'objectif est convexe, l'approche proposée présente les mêmes taux de convergence que l'algorithme Catalyst convexe, entraînant une accélération. Lorsque l'objectif est non-convexe, l'algorithme converge vers les points stationnaires avec le meilleur taux de convergence pour les méthodes de premier ordre. Des résultats expérimentaux prometteurs sont observés en appliquant notre méthode à des problèmes de factorisation de matrice parcimonieuse et à l'entrainement de models de réseaux de neurones.

Abstract

Optimization problems arise naturally in machine learning for supervised problems. A typical example is the empirical risk minimization (ERM) formulation, which aims to find the best a posteriori estimator minimizing the regularized risk on a given dataset. The current challenge is to design efficient optimization algorithms that are able to handle large amounts of data in high-dimensional feature spaces. Classical optimization methods such as the gradient descent algorithm and its accelerated variants are computationally expensive under this setting, because they require to pass through the entire dataset at each evaluation of the gradient. This was the motivation for the recent development of incremental algorithms. By loading a single data point (or a minibatch) for each update, incremental algorithms reduce the computational cost per-iteration, yielding a significant improvement compared to classical methods, both in theory and in practice. A natural question arises: is it possible to further accelerate these incremental methods? We provide a positive answer by introducing several generic acceleration schemes for first-order optimization methods, which is the main contribution of this manuscript.

In chapter 2, we develop a proximal variant of the Finito/MISO algorithm, which is an incremental method originally designed for smooth strongly convex problems. In order to deal with the non-smooth regularization penalty, we modify the update by introducing an additional proximal step. The resulting algorithm enjoys a similar linear convergence rate as the original algorithm, when the problem is strongly convex. More interestingly, a practical optimality certificate based on quadratic lower bounds is systematically given. In contrast to other dual-type algorithms like proximal SDCA, the construction of our algorithm is purely primal. Neither the proof of convergence nor the algorithm requires the Fenchel conjugate, which makes it more practical.

In chapter 3, we introduce a generic acceleration scheme, called Catalyst, for accelerating gradient-based optimization methods in the sense of Nesterov. Our approach applies to a large class of algorithms, including gradient descent, block coordinate descent, incremental algorithms such as SAG, SAGA, SDCA, SVRG, Finito/MISO, and their proximal variants. For all of these methods, we provide acceleration and explicit support for non-strongly convex objectives. The Catalyst algorithm can be viewed as an inexact accelerated proximal point algorithm, applying a given optimization method to approximately compute the proximal operator at each iteration. The key for achieving acceleration is to appropriately choose an inexactness criteria and control the required computational effort. We provide a global complexity analysis and show that acceleration is useful in practice.

In chapter 4, we present another generic approach called QuickeNing, which applies Quasi-Newton principles to accelerate gradient-based optimization methods. The scheme is applicable to the same class of functions as Catalyst, including most

of the well-known first-order algorithms. To the best of our knowledge, QuickeNing is the first Quasi-Newton type algorithm compatible with both composite objectives and the finite sum setting. The algorithm can be interpreted as a combination of inexact L-BFGS algorithm and the Moreau-Yosida regularization. A particular warm start strategy is proposed in order to avoid any line search in the determination of the stepsize. This makes the algorithm practical and we provide extensive experiments showing that QuickeNing gives significant improvement over competing methods in large-scale machine learning problems.

We conclude the thesis by extending the Catalyst algorithm into the nonconvex setting. This is a joint work with Courtney Paquette and Dmitriy Drusvyatskiy, from University of Washington, and my PhD advisors. The strength of the approach lies in the ability of the automatic adaptation to convexity, meaning that no information about the convexity of the objective function is required before running the algorithm. A particular step called Auto-adapt is designed in the algorithm to exploit the local convexity information. When the objective is convex, the proposed approach enjoys the same convergence result as the convex Catalyst algorithm, leading to acceleration. When the objective is nonconvex, it achieves the best known convergence rate to stationary points for first-order methods. Promising experimental results have been observed when applying to sparse matrix factorization problems and neural network models.

Acknowledgements

First of all, I would like to thank my two supervisors Zaid Harchaoui and Julien Mairal for their continuous support and patience during all these years, for offering me a very fascinating research topic, for their enlightening remarks and fruitful advices. I hardly know anything about machine learning when I started my Ph.D, it is with their constant encouragement and meticulous guidance that I am able to come up with this thesis little by little.

I would like to thank François Glineur and Gabriel Peryé, for accepting to review my thesis, and sharing insightful comments and discussions with me. I am also grateful to Jérôme Malick, Alexandre d'Aspremont, Stefanie Jegelka for accepting to be part of the jury; in particular, to Jérôme Malick for being the president of jury and leading the defense.

Furthermore, I would also like to thank my co-authors Courtney Paquette and Dmitriy Drusvyatskiy for many rewarding discussions and their enthusiasm, which results to an accepted paper at AISTATS. I am grateful to Alekh Agarwal and Miro Dudík for offering me the opportunity to visit Microsoft New York. I would also like to express my gratitude to Francis Bach who recommended me to contact Julien and Zaid in the first place. Many thanks to Lear/Thoth Team for providing an incredible research atmosphere and creating lot of fun, to the team leader Cordelia Schmid; to Nathalie Gillot for helping me all the administrative tasks; to my office mates Yang Hua, Erwan Le Roux, Henrique Morimitsu, Jérôme Revaud, Pavel Tokmakov; to the sportive colleagues enjoying soccer Alberto Bietti, Nicolas Chesneau, Guosheng Hu, Dan Oneata, Mattis Paulin, Federico Pierucci, Philippe Weinzaepfel; and to all lovely friends and colleagues that I interacted with, Karteek Alahari, Dexiong Chen, Matthijs Douze, Ghislain Durif, Nikita Dvornik, Thomas Lucas, Vicky Kalogeiton, Xavier Martin, Xiaojiang Peng, Danila Potapov, Gregory Rogez, Shreyas Saxena, Konstantin Shmelkov, Vladyslav Sydorov, Jakob Verbeek, Daan Wynen.

Last but not least, I can not express how grateful I am to my family. In particular, to my mom and my dad for their continued encouragement and support, to my newly married wife, Mengyue Wang, for her understanding and company during the past years. Finally, I would like to dedicate this thesis to my grandfather Rongbao Xu and to loving memory of my grandmother FengYing Ruan, who brought me up with their deep, unconditional love.

Contents

Chapter 1

Introduction

Mathematical optimization lies at the heart of machine learning. The first learning algorithm, Rosenblatt's Perceptron [Duda et al., 2000], was performing incremental optimization on the empirical risk objective with a linear hinge loss for supervised binary classification [Bottou, 2012]. The Adaline algorithm similarly performed incremental minimization on the empirical risk with a least-squares loss function. Incremental and stochastic optimization algorithms were the prevailing approaches to train learning methods in the early days of the computing era [Tsypkin and Nikolic, 1971, Tsypkin, 1973] where datasets were small by today's standards but the computational resources were very limited.

The need to train machine learning models on huge datasets commonly arises in modern applications of machine learning such as visual object recognition and natural language processing, and has recently led to revisit these incremental optimization algorithms with a new point of view. Indeed recent fundamental advances in mathematical optimization allow to provably automatically accelerate first-order or gradient-based optimization methods under broad smoothness assumptions of the objective. The fast gradient method [Nesterov, 1983] and its popular variant FISTA [Beck and Teboulle, 2009] are prominent examples of accelerated first-order methods. These methods were successfully applied to an array of machine learning, signal and image processing, statistics problems; see *e.g.* [Mallat, 2008, Bach et al., 2012, Mairal et al., 2014, Hastie et al., 2015.].

We present in this thesis several acceleration schemes allowing to improve the convergence of first-order optimization algorithms applied to large-scale machine learning problems. We first consider the minimization of convex composite objectives, decomposable into a smooth part and a non-smooth part, using first-order methods. In Chapter 2, we present an incremental optimization algorithm, called Prox-MISO, based on the principle of surrogate minimization and establish its convergence guarantee. In Chapter 3, we describe an acceleration scheme called Catalyst consisting of an outer-loop performing an extrapolation step on iterates produced by first-order methods applied to auxiliary objectives. We quantify the acceleration in terms of worst-case non-asymptotic convergence guarantees when applying Catalyst to various first-order methods. In Chapter 4, we describe a different acceleration scheme called QuickeNing also

based on an inner-outer loop construction but performing quasi-Newton-type updates on the iterates produced by first-order methods applied to auxiliary objectives. We demonstrate the effectiveness of the acceleration through extensive numerical benchmarks on several machine learning datasets. Finally, in Chapter 5, we consider the minimization of possibly non-convex composite objectives. We present an extension of the Catalyst scheme that is automatically adaptive to the hidden convexity of the problem.

In this introductory chapter, we shall review the main concepts at work in the acceleration schemes we propose. We first formulate the optimization problem and review two important classes of smooth convex optimization methods: first-order methods and Quasi-Newton methods. Then we review the notion of proximal operator in Section 1.4, which is an important tool for extending smooth optimization methods to non-smooth settings. In Section 1.5, we present a survey of incremental optimization algorithms, which are particularly attractive when dealing with datasets with large number of data-points. A natural question to ask is can we do better? To address this question, we introduce the Moreau-Yosida regularization in Section 1.6, which gives a natural interpretation of the generic acceleration scheme that we develop in this thesis. Finally, we outline the technical contributions in Section 1.7.

1.1 Formulation of the optimization problem

The goal of machine learning is to have a machine learn from data by itself without human interventions. Often a specific task is given, such as visual object recognition or speech recognition, and we would like to ask machines to solve the corresponding task based on the data we provided. For example, in the context of visual object recognition, we would like to learn a function that maps images to an object label, indicating what object category appears in the image, such as a cat, a dog; or a plane, etc. However, the set of functions from images to labels are so rich that given any finite many images, there will always be a perfect match by simply memorizing the correct label of each image. This perfect matching is contrary to our intention of learning high-level representations of the image, because it may behaves arbitrarily bad for any unseen image. Thus it is necessary and important to impose some hypothesis on the nature of the function that we want to learn in order to get a good generalization ability [Vapnik, 2013]. Different models encode different assumption about the function space: in linear model, the function class is simply all linear functions; in more complex model like neural networks, the function space is a collection of connected artificial neurons mimicking human brain's structure [Friedman et al., 2001]. Thus the learning model presumes the underlying structure of the problem and provides a much smaller searching space of functions. Once we determine which model to be applied, we can formulate the problem as an optimization problem among the providing searching space.

More formally speaking, we are given two spaces of objects $\mathcal{A}$ and $\mathcal{B}$ and we assume that there is a joint probability distribution $P(a, b)$ over $\mathcal{A} \times \mathcal{B}$. The goal is to learn a function $h : \mathcal{A} \to \mathcal{B}$ predicting the label in $\mathcal{B}$ given an object $a \in \mathcal{A}$ under a certain predefined searching

space $h \in \mathcal{H}$. The quality of the prediction function h is measured by the risk,

$$R(h) = \mathbb{E}_{P(a,b)}[\ell(h(a), b)],$$

where ℓ is a distance function measuring the difference between the predicted label $h(a)$ and the true label b. Ideally, we would like to find the best prediction function minimizing the risk which gives the optimization problem

$$\min_{h \in \mathcal{H}} R(h). \tag{1.1}$$

However, the latent distribution P is unknown and therefore the risk function R is directly uncomputable. Instead, we assume that n i.i.d samples $(a_i, b_i)_{i \in [1,n]}$ from P are provided, and we define the empirical risk with respect to this training set as

$$R_{\text{emp}}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(a_i), b_i).$$

By the law of large numbers, the empirical risk converges to the expected risk when the number of samples n goes to infinity. Thus, we minimize the empirical risk among the class of functions defined by the model, which forms the *empirical risk minimization problem* (ERM) [Shalev-Shwartz and Ben-David, 2014]. When the model is parametric, this naturally transforms to an optimization problem

$$\min_{x \in \mathbb{R}^p} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(h(a_i, x), b_i) \right\}, \tag{1.2}$$

where x represents the model parameters. In linear regression, x corresponds to the coefficient of the linear representation $a_i^T x$; in neural networks, x contains all the layers' weights. Typical examples of loss functions are the square loss for least squares regression, *i.e.*, $\ell(h, b) = (b - h)^2$ with $b \in \mathbb{R}$ and the logistic loss $\ell(h, b) = \log(1 + \exp(-bh))$ for logistic regression with $b \in \{-1, +1\}$. As a consequence, the learning problem in the setting of supervised learning turns out to be an optimization problem, which motivates our work towards fast and efficient algorithms to solve the ERM problems.

1.2 First-order methods for smooth optimization

For the sake of clarity, we write the empirical risk minimization problem in the general form

$$\min_{x \in \mathbb{R}^p} f(x) \ . \tag{1.3}$$

Convexity and smoothness. We shall first focus on the case where f is **smooth and convex**, which includes classical models such as linear regression and logistic regression. We assume the smoothness assumption at this first stage only for the sake of clarity, it allow us to define the gradient which is a natural descent direction. The extension to composite function settings will be discussed later in Section 1.4.

Definition 1. *A continuously differentiable function f is called **convex** if for any x, y in $\mathbb{R}^p$, we have*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle, \tag{1.4}$$

where $\langle \cdot, \cdot \rangle$ denotes the canonical Euclidean inner product of $\mathbb{R}^p$.

If, in addition, inequality (1.4) can be improved with an additional quadratic term, namely there exists $\mu > 0$ such that for any x, y in $\mathbb{R}^p$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2, \tag{1.5}$$

*then we say that the function f is **μ-strongly convex**, where μ is the strong convexity parameter.*

Convexity guarantees that all stationary points of f are in fact global minima. The strong convexity further guarantees that a global minimum exists and is unique. When f is two times continuously differentiable, the strong convexity assumption is equivalent to saying that the eigenvalues of the Hessian $\nabla^2 f(x)$ are uniformly lower bounded by $\mu > 0$ for all x in $\mathbb{R}^p$. Intuitively, a strongly convex function enjoys better local curvature and steeper slope, which allows optimization algorithms to converge faster.

As we can see, Equation (1.4) and (1.5) provide simple lower bounds on f. In the same spirit, we also assume that an upper bound is available, leading to the definition of L-smoothness. We say that f is **L-smooth** if f is differentiable and its gradient is L-Lipschitz, i.e. for any x, y in $\mathbb{R}^p$

$$\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|. \tag{1.6}$$

Elementary calculation shows that this condition implies that for any x, y in $\mathbb{R}^p$,

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2, \tag{1.7}$$

which provides a quadratic upper bound; see details in Theorem 2.1.5 of [Nesterov, 2004]. Moreover, when f is two times continuously differentiable, L-smoothness is equivalent to having the eigenvalues of the Hessian $\nabla^2 f(x)$ uniformly upper bounded by L, for any x in $\mathbb{R}^p$. To summarize, L and μ represent upper and lower bounds on the eigenvalues of the Hessian respectively, which provide simple quadratic upper and lower bounds on f. This brings us to the **condition number** of the function f, defined as

$$Q_f = \frac{L}{\mu}, \tag{1.8}$$

which plays an important role in the characterization of the convergence rate of first-order algorithms applied to f.

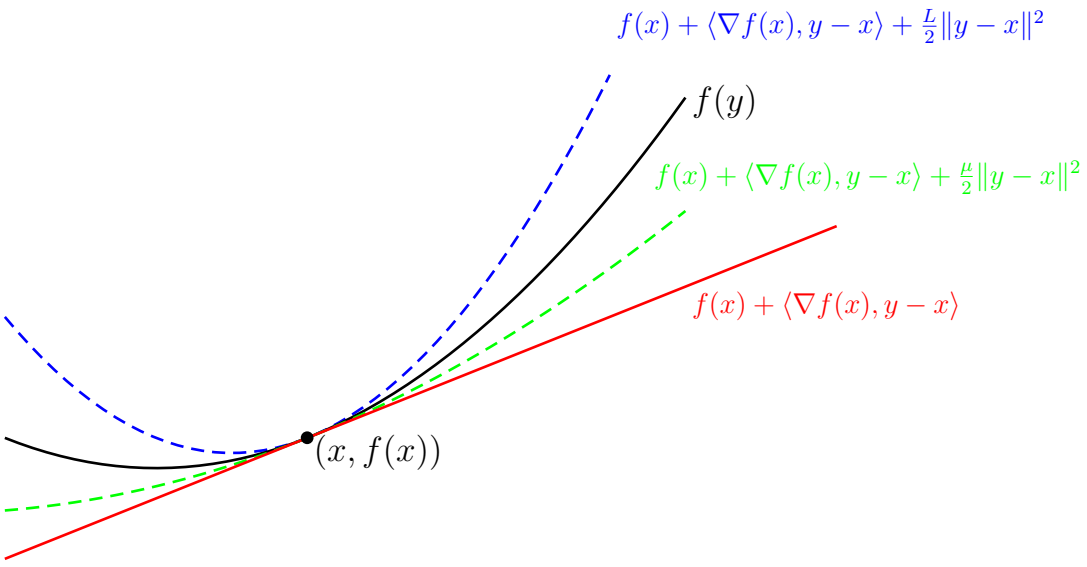

Figure 1.1 – Graphical illustration of L-smoothness and μ-strong convexity.

1.2.1 Gradient descent algorithm

We come back to our optimization problem (1.3). The goal is to design algorithms which move towards the minimum iteratively. Given that we are at point x, a natural idea is to follow the direction giving the steepest decrease in terms of the function value. Since the objective function f is differentiable, given any unit direction u, the local variation of function f around x following the direction u is given by

$$df_x(u) = \nabla f(x)^\top u.$$

Thus, the direction in which the function decreases the most is the direction of anti gradient $-\nabla f(x)$. This yields the classical gradient descent algorithm:

Gradient Descent algorithm
Initialization: $x_0 \in \mathbb{R}^d$.
For $k \geq 0$, $$x_{k+1} = x_k - \eta_k \nabla f(x_k),$$ where $\eta_k > 0$ is the stepsize.

There are several strategies to set the sequence of stepsizes $(\eta_k)_{k\geq 0}$. A simple strategy is to choose a constant stepsize $\eta_k = \eta$ for all $k \geq 0$, which enjoys the following convergence guarantee.

Theorem 1 (**Convergence of gradient descent method**). *If f is convex and L-smooth, then by letting $\eta_k = \frac{1}{L}$, the sequence $(x_k)_{k\geq 0}$ generated by the gradient descent algorithm satisfies*

$$f(x_k) - f^* \leq \frac{L\|x_0 - x^*\|^2}{2k}, \quad \forall k \geq 1, \tag{1.9}$$

where f^* is the minimum of f and x^* is an optimal point with $f(x^*) = f^*$. Moreover, if f is μ-strongly convex, then

$$f(x_k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f^*)\,. \tag{1.10}$$

The proof is recalled in Appendix A.1. It is worth mentioning that the convergence rate shown in (1.9) is not tight, an exact worst case convergence rate for gradient methods has been recently derived by Taylor et al. [2017], De Klerk et al. [2017]. More precisely, they show that

$$\|x_k - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)^{2k} \|x_0 - x^*\|^2, \tag{1.11}$$

$$f(x_k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^{2k} (f(x_0) - f^*)\,, \tag{1.12}$$

$$\text{and} \quad \|\nabla f(x_k)\|^2 \leq \left(1 - \frac{\mu}{L}\right)^{2k} \|\nabla f(x_0)\|^2, \tag{1.13}$$

when applying gradient descent method with stepsize $\frac{1}{L}$ on a μ-strongly convex function. The proof is elementary and we refer readers who are interested to the original papers.

The convergence rate in Theorem 1 is stated respect to the function value gap, one can directly derive from it that the norm of the gradient $\|\nabla f(x_k)\|$ vanishes by using Theorem 2.1.5 of Nesterov [2004]:

$$\frac{1}{2L}\|\nabla f(x_k)\|^2 \leq f(x_k) - f^*.$$

The quantity $\|\nabla f(x_k)\|$ may be naturally checked in practice and it provides information about convergence to a stationary point. So far, we have been considering convex objective functions, for which all stationary points are also global minima. Therefore, for convex objective functions, we are able to state stronger convergence guarantees in terms of function evaluations $f(x_k) - f^*$. However, when the function becomes non-convex, we cannot hope to converge to a global minimum, and as a substitute, we study the convergence to a stationary point which is characterized by the gradient norm.

As shown by the theorem, the asymptotic convergence rate of the gradient descent algorithm is linear when the problem is strongly convex, which is much faster compared to the rate for non-strongly convex problems:

Definition 2. *We say that an algorithm $\mathcal{M}$ is **linearly convergent** on a problem f if there exists $\tau_{\mathcal{M}}$ in $(0, 1]$, $C_{\mathcal{M}} > 0$ such that for any initial point x_0 in $\mathbb{R}^p$, $\mathcal{M}$ generates a sequence of iterates $(x_k)_{k\geq 0}$ satisfying*

$$f(x_k) - f^* \leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^k \Delta(x_0), \tag{1.14}$$

*where $\Delta(x_0)$ is a quantity depending on x_0 measuring how good the initial point x_0 is. We will call $\tau_{\mathcal{M}}$ the **linear convergence parameter** of the algorithm $\mathcal{M}$.*

Note that the asymptotic behavior of the algorithm does not change as long as $\Delta(x_0)$ is finite, which is why we leave it unspecified on purpose. Typical examples are $f(x_0) - f^*$, $\|x_0 - x^*\|^2$ or some kind of duality gap, depending on the considered algorithm. Unlike $\Delta(x_0)$, $\tau_{\mathcal{M}}$ and $C_{\mathcal{M}}$ are quantities that do not depend on the initial point but on the algorithm $\mathcal{M}$ and the problem f itself. As we can see, for gradient descent, $C_{\mathcal{M}} = 1$ and $\tau_{\mathcal{M}} = \mu/L$, which is the inverse of the condition number of f. Consequently, the asymptotic behavior of gradient descent depends only on the condition number. The larger it is, the slower the algorithm converges, in which case the problem is called **ill-conditioned**.

To give an example of this phenomenon, let us consider a quadratic objective function $f(x) = \frac{1}{2}x^T Dx$, where D is a diagonal matrix with coefficients $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > 0$. The Hessian of f is then constant: $\nabla^2 f(x) = D$ for any $x \in \mathbb{R}^p$. Consequently, we have $L = \sigma_1$, $\mu = \sigma_d$, and the condition number $Q = \sigma_1/\sigma_d$. Now let us see how the gradient descent algorithm solves this problem. Given an initial point x_0, one step of gradient descent gives

$$x_1 = x_0 - \frac{1}{\sigma_1}\nabla f(x_0) = \left(I - \frac{1}{\sigma_1}D\right) x_0.$$

Consequently, by induction, we have

$$x_k = \left(I - \frac{1}{\sigma_1}D\right)^k x_0 \;\longrightarrow\; x^* = 0.$$

The rate of convergence of $\|x_k - x^*\|$ is asymptotically controlled by the last coordinate of the vector, which is

$$x_k^{(d)} = \left(1 - \frac{\sigma_d}{\sigma_1}\right)^k x_0^{(d)} = \left(1 - \frac{\mu}{L}\right)^k x_0^{(d)}.$$

This matches the convergence rate in terms of distance to the optimum shown in (1.11) and one can easily derive similar result in terms of function value gap. In the best case when $L = \mu$, the quadratic function is isotropic and one step of gradient descent at any arbitrary point attains the minimum. Intuitively, the smaller the condition number is, the more isotropic the objective function is, which makes it more favorable to the gradient descent algorithm.

For later comparison of the computational complexity, we translate our convergence result in terms of numbers of gradient evaluations, which equals to the number of iterations in the gradient descent algorithm:

Corollary 1. *Let $\varepsilon > 0$, the number of iterations N required to achieve an ε-solution $f(x) - f^* \leq \varepsilon$ by the gradient descent method is*

$$\begin{cases} O\left(\frac{L}{\varepsilon}\right) & \text{if } f \text{ is convex,} \\ O(Q_f \log\left(\frac{1}{\varepsilon}\right)) & \text{if } f \text{ is } \mu\text{-strongly convex.} \end{cases} \tag{1.15}$$

There are other, sometimes more effective, strategies to set the sequence of stepsizes $(\eta_k)_{k\geq 0}$ [Nocedal and Wright, 2006, Bonnans et al., 2006]. A standard technique is to perform a line search. The exact line search strategy is described as follows: at each iteration k, we choose η_k which gives the steepest decrease, namely

$$\eta_k = \arg\min_{\eta>0} f(x_k - \eta \nabla f(x_k)).$$

Such a strategy is aggressive since we always take the best step-size possible given the direction of gradient. However, the exact stepsize is often not computable. A more practical way is to apply a backtracking line search strategy, that looks for a stepsize η_k such that the following descent condition is satisfied,

$$f(x_k - \eta_k \nabla f(x_k)) \leq f(x_k) - \frac{\eta_k}{2} \|\nabla f(x_k)\|^2.$$

When the inequality is violated, we decrease the stepsize η_k by a constant factor and recheck it again. Note that the descent condition is satisfied for any stepsize $\eta_k < \frac{1}{L}$, thus the line search procedure is guaranteed to terminate. In terms of convergence result, the backtracking line search enjoys a similar asymptotic convergence rate as the constant step-size, see section 9.2 of [Boyd and Vandenberghe, 2009]. Thus for later discussion, by gradient descent algorithm, we refer to the one with constant step-size [Bertsekas, 2015].

1.2.2 Accelerated gradient method

A natural question is whether or not the gradient descent method is optimal. The answer is provided in part by the concept of minimax lower bound analysis, which indicates the best possible rate of convergence expected for a given class of algorithms. We constrain ourselves on first-order methods because problems from machine learning potentially contain high-dimensional features, where the evaluation of higher-order differentials is infeasible in practice due to memory issues and prohibitive computational effort. Let us now introduce the notion of the iterative first-order method [Nesterov, 2004]:

Assumption 1. *We call an algorithm $\mathcal{M}$ an iterative first-order method if it generates a sequence of iterates $(x_k)_{k\geq 0}$ such that*

$$x_k \in x_0 + \mathit{Span}\{\nabla f(x_0), \cdots, \nabla f(x_{k-1})\}, \quad \text{for } k \geq 1.$$

This assumption may seem a bit restrictive because we constrain the trajectory to be a linear combination of past gradients, but we remark that most interesting first-order algorithms satisfy this definition including the gradient descent algorithm, its accelerated variants, conjugate gradient methods and quasi-Newton algorithms such as BFGS and L-BFGS algorithms.[1] We shall therefore recall the minimax lower bounds for first-order methods as defined by Assumption 1.

Theorem 2 (**Lower bounds for convex functions [Nemirovskii et al., 1983, Nesterov, 2004]**). *Given the dimension d, for any k with $1 \leq k \leq \frac{1}{2}(d-1)$, and any x_0 in $\mathbb{R}^p$, there exists a convex L-smooth function f such that for any first-order method $\mathcal{M}$ satisfying Assumption 1,*

$$f(x_k) - f^* \geq \frac{3L\|x_0 - x^*\|^2}{32(k+1)^2},$$
$$\|x_k - x^*\|^2 \geq \frac{1}{8}\|x_0 - x^*\|^2.$$

Note that the above result is valid only when the number of iterations is bounded by half of the dimension, which does not provide any information about the asymptotic convergence. A dimension free lower bound in a similar rate was recently developed by Arjevani and Shamir [2016] relying on a stronger assumption, which excludes quasi-Newton methods.

Theorem 3 (**Lower bounds for strongly convex functions [Nemirovskii et al., 1983, Nesterov, 2004]**). *Consider the space of square-summable sequences $\ell_2(\mathbb{R})$. For any x_0 in $\ell_2(\mathbb{R})$, any constant $\mu > 0$, $Q > 1$, there exists a μ-strongly convex function f with condition number Q such that for any first-order method $\mathcal{M}$ satisfying Assumption 1, we have,*

$$\|x_k - x^*\|^2 \geq \left(1 - \frac{2}{\sqrt{Q}+1}\right)^{2k} \|x_0 - x^*\|^2,$$
$$\text{and} \quad f(x_k) - f^* \geq \frac{\mu}{2}\left(1 - \frac{2}{\sqrt{Q}+1}\right)^{2k} \|x_0 - x^*\|^2, \quad \text{for any} \quad k \geq 1.$$

As we can see, there is a gap between the lower bounds and the rate of convergence obtained from the gradient descent algorithm, which gives hope for algorithms with faster rate. In 1983, Nesterov proposed an accelerated version of gradient method, which is optimal with respect to the minimax lower bound.

[1] See Proposition 18 in the Appendix.

Accelerated Gradient method

input $x_0 \in \mathbb{R}^d$ and $\alpha_0 \in (0,1)$.

1: Set $y_0 = x_0$ and $q = \frac{\mu}{L}$.

2: At k-th iteration, $(k \geq 0)$

(a) Compute $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$.

(b) Compute $\alpha_{k+1} \in (0,1)$ from equation

$$\alpha_{k+1}^2 = (1-\alpha_{k+1})\alpha_k^2 + q\alpha_{k+1}.$$

(c) Compute y_k with

$$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k) \quad \text{with} \quad \beta_k = \frac{\alpha_k(1-\alpha_k)}{\alpha_k^2 + \alpha_{k+1}}.$$

The key ingredient for acceleration is to take the gradient at the extrapolated point y_k instead of x_k, where y_k is a linear combination (but not necessarily an average) of the previous iterates. Nesterov's acceleration falls into the general class of schemes called extrapolation schemes in numerical analysis. Intuitively, y_k moves one step further from x_k following the "descent direction" $x_k - x_{k-1}$, yielding faster convergence. A lot of work has been devoted to provide an intuitive explanation of the magic behind Nesterov's acceleration scheme. For instance, y_k can be viewed as a linear combination of a gradient descent step and a mirror descent step [Allen-Zhu and Orecchia, 2014]; or geometrically, the algorithm can be interpreted as sequentially performing the intersection of balls [Bubeck et al., 2015] or sequentially averaging some quadratic functions [Drusvyatskiy et al., 2016]; or from a continuous point of view, it can be seen as applying a multi-step integration schemes on the gradient flow equation [Scieur et al., 2017]. The important message is that by simply taking the gradient step at a different point, we can achieve the optimal rate of convergence:

Theorem 4 (Convergence of accelerated gradient method [Nesterov, 2004]). *If f is convex, the accelerated gradient algorithm generates a sequence $(x_k)_{k\geq 0}$ such that*

$$f(x_k) - f^* \leq \frac{4L}{(k+2)^2}\|x_0 - x^*\|^2.$$

Moreover, when f is μ-strongly convex, we have

$$f(x_k) - f^* \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|x_0 - x^*\|^2.$$

Corollary 2. *Let $\varepsilon > 0$, the number of iterations N required to achieve an ε-solution $f(x) - f^* \leq \varepsilon$ by the accelerated gradient descent is*

$$\begin{cases} O\left(\sqrt{\frac{L}{\varepsilon}}\right) & \text{if } f \text{ is convex,} \\ O(\sqrt{Q}\log\left(\frac{1}{\varepsilon}\right)) & \text{if } f \text{ is } \mu\text{-strongly convex.} \end{cases} \tag{1.16}$$

Due to the extrapolation step, the accelerated gradient method is not guaranteed to be a descent algorithm, meaning that it is possible to have $f(x_{k+1}) > f(x_k)$. One way to ensure the descent property is to introduce a restart scheme, which restarts the algorithm once some descent conditions are violated [O'donoghue and Candes, 2015, Fercoq and Qu, 2016]. This forces the algorithm to make progress at each iteration and achieves significant improvement in practice. To summarize, the accelerated gradient descent algorithm improves the convergence rate compared to standard gradient methods and achieves the optimal minimax rate. However, as mentioned before, such a minimax rate only holds when the number of iteration is small, there may be hope to achieve even faster algorithms as the number of iterations increases. As we will see later, under slightly stronger assumption, quasi-Newton algorithms can asymptotically achieve superlinear convergence.[2]

1.3 Quasi-Newton methods

We have focused so far on first-order optimization algorithms. If the objective function is smooth enough, using higher order information may result in faster algorithms. Newton's method is one of the most successful optimization method combining first and second derivatives. Instead of performing the first-order approximation of the function, Newton's method goes one step further. It uses the second order Taylor expansion to approximate the objective function,

$$f(x_k + h) \approx f(x_k) + \nabla f(x_k)^T h + \frac{1}{2} h^T \nabla^2 f(x_k) h. \tag{1.17}$$

Minimizing this quadratic function leads to the update

$$x_{k+1} = x_k - \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k). \tag{1.18}$$

When the objective is quadratic, one step of Newton's method gives the exact minimum. In general, when $(x_k)_{k\geq 0}$ converges and the minimum has a nondegenerate Hessian, the sequence converges with a quadratic rate. However, for a general nonlinear function, the iterate (1.18) is not guaranteed to converge for any arbitrary initial point x_0. This is because the step $(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$ may be large and in which case the Taylor expansion (1.17) may no longer

[2] Meaning $\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \to 0$.

be a good approximation of f. A quick solution to guarantee the convergence is to introduce a stepsize parameter $\eta_k > 0$ in the update,

$$x_{k+1} = x_k - \eta_k \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k), \tag{1.19}$$

and ensure that the function value is decreasing, *e.g.* $f(x_{k+1}) < f(x_k)$. Such stepsize exists as long as the Hessian at x_k is positive definite.

While Newton's method enjoys a fast convergence rate, it is computationally expensive. At each iteration, the Hessian matrix needs to be evaluated and to be inverted, requiring at least an $O(d^3)$ computational effort. Thus, the per-iteration cost of Newton's method is d^2 times higher than the gradient methods, which is often too expensive for solving large-scale machine learning problems. This is why Hessian-free and inverse-free modifications of Newton's method have been developed, giving birth to the class of Quasi-Newton methods, defined as the generalization of (1.18):

$$x_{k+1} = x_k - B_k^{-1} \nabla f(x_k), \tag{1.20}$$

The matrix B_k represents an approximation of the Hessian matrix, varying from iteration to iteration, giving the alternative name: variable metric methods. The update in (1.20) can be viewed as performing a gradient descent step according to the metric induced by B_k. The metrics B_k are generated iteratively without evaluating the Hessian, and its inverse $H_k = B_k^{-1}$ is given in a closed-form update, without any operation of matrix inversion. This makes Quasi-Newton algorithm computationally efficient. For historical reasons, we use B to denote approximations of the Hessian matrix and H to denote the inverse of the Hessian matrix.

The challenging point of Quasi-Newton algorithms lies in the construction of the Hessian approximations. In a few words, Quasi-Newton methods use the finite difference of the gradient to approximate the Hessian matrix (see Appendix C), which gives a fundamental equation called the secant equation: at $(k+1)$-th iteration, the Hessian approximation B_{k+1} is required to satisfy the relationship

$$B_{k+1} s_k = y_k, \quad \text{with} \quad s_k = x_{k+1} - x_k, \; y_k = \nabla f(x_{k+1}) - \nabla f(x_k). \tag{1.21}$$

Intuitively, the function f is locally approximated by the quadratic function

$$Q_{k+1}(x) = f(x_{k+1}) + \nabla f(x_{k+1})^T (x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T B_{k+1} (x - x_{k+1}).$$

The secant equation (1.21) is requiring the gradient ∇f to match the gradient of the quadratic approximation ∇Q_{k+1} at the past iterate x_k. Moreover, since the Hessian matrix is always symmetric, and positive definite if the function is strongly convex, it is natural to ask B_{k+1} to have the same properties. Thus, the goal is to construct a symmetric and positive definite matrix B_{k+1} satisfying the secant equation. However, this is an underdetermined problem based on the local estimation. In order to capture the past information, we would expect B_{k+1} to be "close" to B_k, yielding the two most popular Quasi-Newton algorithms called *Davidon-Fletcher-Powell*

(DFP) update [Davidon, 1991, Fletcher and Powell, 1963] and *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) update [Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970].

In the DFP update, the matrix B_{k+1} is defined by

$$B_{k+1}^{\text{DFP}} = \left(I - \frac{y_k s_k^T}{\langle y_k, s_k\rangle}\right) B_k \left(I - \frac{s_k y_k^T}{\langle y_k, s_k\rangle}\right) + \frac{y_k y_k^T}{\langle y_k, s_k\rangle}, \tag{1.22}$$

and the corresponding inverse matrix is obtained by applying the Sherman-Morrison formula [Sherman and Morrison, 1950], giving

$$H_{k+1}^{\text{DFP}} = H_k + \frac{s_k s_k^T}{\langle s_k, y_k\rangle} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}. \tag{1.23}$$

In the BFGS update, the inverse Hessian approximation H_{k+1} is directly updated by

$$H_{k+1}^{\text{BFGS}} = \left(I - \frac{s_k y_k^T}{\langle y_k, s_k\rangle}\right) H_k \left(I - \frac{y_k s_k^T}{\langle y_k, s_k\rangle}\right) + \frac{s_k s_k^T}{\langle y_k, s_k\rangle}. \tag{1.24}$$

A perfect symmetry can be observed from (1.22) and (1.24) by interchanging $s_k \leftrightarrow y_k$ and $B_k \leftrightarrow H_k$. In fact, the underlying mechanism of the two algorithms are identical, where DFP approximates the Hessian matrix and BFGS approximates the inverse Hessian matrix. This is why BFGS is sometimes named the complementary DFP algorithm.

Quasi-Newton Method: DFP/ BFGS

input $x_0 \in \mathbb{R}^p$ and $H_0 \in \mathbb{R}^{d\times d}$.

1: At $(k+1)$-th iteration, $(k \geq 0)$

(a) Compute the iterate

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

where the stepsize η_k is determined by a line-search strategy or constant.

(b) Update H_{k+1} with $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$:

$$\begin{aligned} H_{k+1}^{\text{DFP}} &= H_k + \frac{s_k s_k^T}{\langle s_k, y_k\rangle} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}; \\ H_{k+1}^{\text{BFGS}} &= \left(I - \frac{s_k y_k^T}{\langle y_k, s_k\rangle}\right) H_k \left(I - \frac{y_k s_k^T}{\langle y_k, s_k\rangle}\right) + \frac{s_k s_k^T}{\langle y_k, s_k\rangle}. \end{aligned}$$

Both DFP and BFGS algorithms inherit a nice property of Newton's method, which is the affine invariance. More precisely, when considering a new problem $g(z) = f(Az)$, where A in $\mathbb{R}^{d\times d}$ is a non singular matrix. Applying DFP and BFGS algorithm to the problem $\min_z g(z)$ provide identical updates as the problem $\min_x f(x)$, if the initialization metric satisfies the affine invariance relation $H_0^f = AH_0^g A^T$. This suggests that the convergence of Quasi-Newton methods are independent of the condition number if we initialize them with an exact Hessian matrix. However, in practice, H_0 is usually set to be proportional to the identity matrix, which breaks the affine invariance property.

In order to guarantee convergence, the step size of Quasi-Newton updates need to be carefully chosen. A typical example is to perform a line-search:

(a) **Exact line-search:** Given a direction d, the step-size η is chosen to minimize the function in this direction:

$$\eta_{\text{exact}} = \arg\min_{\eta} f(x + \eta d).$$

(b) **Wolfe's conditions:** Given a direction d, two scalars $0 < \alpha < \beta < 1$, the setpsize η is chosen to satisfy

$$f(x + \eta d) \leq f(x) + \alpha\eta\nabla f(x)^T d, \qquad \text{(Armijo condition)}$$
$$\nabla f(x + \eta d)^T d \geq \beta\nabla f(x)^T d. \qquad \text{(curvature condition)}$$

Note that the exact line-search is computationally expensive in practice, but it provides interesting theoretical insight. The convergence of the DFP or BFGS algorithms depends on the selected line-search strategy.

Theorem 5. *Assume that the objective function f is twice differentiable, L-smooth and μ-strongly convex.*

1. *If f is quadratic and the exact line search is used, both DFP and BFGS find the exact minimum in at most d iterations [Broyden, 1967].*

2. *For a general function f, if the exact line search is used, then both DFP and BFGS algorithms converge linearly [Powell, 1971]. Furthermore, if the Hessian is Lipschitz, then both algorithms converge superlinearly, meaning that*

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \to 0.$$

3. *If the stepsize is always equal to 1, and the initialization is close enough to the solution, then both DFP and BFGS algorithms converge linearly. Furthermore, if the Hessian is Lipschitz, then both algorithms converge superlinearly. [Dennis and Moré, 1974]*

4. *If the stepsize is determined by Wolfe's conditions, then the BFGS algorithm converges linearly. Furthermore, if the Hessian is Lipschitz, then the algorithm converges superlinearly. [Byrd et al., 1987, Byrd and Nocedal, 1989]*

We remark that when equipped with the exact line-search, both DFP and BFGS algorithms are closely related to the conjugate gradient method, which exhibits a finite termination when the objective function is quadratic. Moreover, a global convergence result is proved for both methods using the exact line-search. However the scenario becomes quite different when a practical strategy is used, like a constant step-size or Wolfe's conditions. The DFP algorithm only enjoys local linear convergence while the BFGS algorithm achieves global convergence. An example is given in [Powell, 1986] illustrating such a different behavior while using the constant step-size $\eta_k = 1$. The objective function is a quadratic function, but the initialization B_0 is far from the true Hessian with an extremely large eigenvalue λ. The BFGS algorithm manages to correct it in a linear rate, namely $\lambda_{k+1} \approx \lambda_k/2$; but the DFP algorithm reduces it at a much slower rate, giving an update $\lambda_{k+1} \approx \lambda_k - 1$, see details in [Powell, 1986]. Therefore, it may take a large amount of iterations for the DFP algorithm to get into the region with linear convergence. This might explain why the DFP algorithm does not perform as well as the BFGS algorithm in practice. Thus, we will focus on the BFGS algorithm in the following discussion. In terms of convergence rate, the theoretical rate of BFGS algorithm with Wolfe's conditions is not better than the gradient descent algorithm, requiring $O\left(\frac{L}{\mu}\log(\frac{1}{\varepsilon})\right)$ iterations to achieve an ε-solution. On contrary, in practice, it is often observed that the BFGS algorithm significantly outperforms the gradient descent algorithm and its accelerated variant. Moreover, all the Quasi-Newton methods we mention are first-order algorithms satisfying Assumption 1 if the initialization B_0 or H_0 is proportional to the identity matrix. To verify this, it suffices to decompose the matrix-vector product $H_k\nabla f(x_k)$ into a weighted sum of previous gradients; see Proposition 18 in the Appendix for more details.

When comparing to classical gradient methods, the sole additional ingredient used by Quasi-Newton methods is to maintain a Hessian approximation at each iteration. However, in the large-scale setting, the required d^2 space may still be too large to fit into memory. This point motivated the study of limited memory Quasi-Newtons methods, giving birth to the *limited memory BFGS algorithm* (L-BFGS) [Liu and Nocedal, 1989]. The main idea is to use the decomposition of matrix-vector product mentioned before, where the update $x_{k+1} - x_k$ can be expressed as a weighted sum of the previous gradients. Thus, instead of keeping in memory the approximate Hessian matrix, one can replace it by a list of past gradients. The trick to limit the memory is to fix the maximum length ℓ for such a list and only keep the most recent gradients. As a consequence, the required memory is reduced from d^2 to ℓd where ℓ is selected by the user. In order to compute the matrix-vector product $H_k\nabla f(x_k)$, a two-loop recursion strategy has been developed, yielding a complexity $O(ld)$ instead of $O(d^2)$ [Nocedal and Wright, 2006]. Thus the L-BFGS algorithm improves the BFGS algorithm both in terms of memory space and computational effort per iteration, which makes it practical. However, the superlinear convergence result is no longer valid since the limited memory may be too small

to well approximate the Hessian matrix. It is shown that L-BFGS enjoys a linear convergence rate when the function is strongly convex; but the rate is again not better than the gradient methods.

As a summary, Quasi-Newton algorithms maintain a Hessian approximation to exploit the curvature of the objective function, a detailed overview with additional algorithms apart from DFP or BFGS is provided in Section C. In practice, the step-size needs to be selected carefully. The line-search using Wolfe's conditions usually provides promising performance [Nocedal and Wright, 2006]. However, in contrast to the significant speed up, the theoretical analysis does not give better convergence results compared to the gradient method, leaving a huge gap between theory and practice.

1.4 Proximal algorithms

Overfitting is a common issue when applying the empirical risk minimization framework. This phenomenon usually occurs when the number of samples is too small or the dimension of the features is too large, reminding that the training set is just a finite sample of the latent distribution. If the training set is too small, it may not reflect the underlying structure of the true distribution and, fitting a model on this dataset will most likely capture some irrelevant artifacts. If the dimension of the features is too large, the model may successfully predict any training data including noise and outliers, which will not generalize well to unseen data.

One solution for preventing overfitting is to add a regularization penalty in the ERM problem (1.2), *e.g.*

$$\min_{x\in\mathbb{R}^p} \left\{ \frac{1}{n}\sum_{i=1}^{n} \ell(h(a_i,x),b_i) + \psi(x) \right\}. \tag{1.25}$$

The purpose of the regularizer ψ is to impose some constraints and prior knowledge about the model parameters. For example, by adding the squared ℓ_2-norm, known as Ridge regularization [Engl et al., 1996], we penalize large entries in the parameter x. If we want to limit the number of selected features, we can apply sparsity-inducing regularizers. A natural candidate is the ℓ_0-penalty defined by

$$\|x\|_0 = \#\{i = 1, \cdots, d \mid x_i \neq 0\}.$$

However, the ℓ_0-penalty is non convex which leads to an NP-hard problem [Natarajan, 1995]. Therefore, to make the problem computationally tractable, we substitute the ℓ_0-penalty by its convex relaxation – the ℓ_1-norm. Regularizing the linear regression with the ℓ_1-norm yields the well known Lasso problem [Tibshirani, 1996]. By forcing the sum of the absolute values of the model parameter to be small, Lasso is able to encourage certain coefficients to be zero and induces sparsity. A number of Lasso variants have been developed later to improve the performance under different scenarios including Elastic-net regularization [Zou and Hastie, 2005], Group-Lasso with or without overlap [Yuan and Lin, 2006, Jacob et al., 2009, Jenatton et al., 2011] and other sparsity-inducing regularizers [Bach et al., 2012].

Another benefit of considering composite minimization is that it covers convex constrained minimization problems. This is achieved by defining ψ as the extended-valued indicator function $\mathbb{1}_{\mathcal{C}}$ of a convex domain $\mathcal{C}$, which takes the value 0 inside the convex set $\mathcal{C}$ and $+\infty$ outside (see [Hiriart-Urruty and Lemaréchal, 1996]). Thus, we consider the following composite optimization problem

$$\min_{x\in\mathbb{R}^p} \left\{ f(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \psi(x) \right\} \tag{1.26}$$

where each f_i is convex, L-smooth and ψ is convex, but not necessarily differentiable. In the following, we will denote f_0 to be the smooth part of the objective, namely $f_0 = \frac{1}{n}\sum_{i=1}^{n} f_i$.

A side effect of the regularization is that the objective function may be no longer differentiable, which is problematic to apply smooth optimization algorithms. In order to handle this issue, we introduce the notion of proximal operator:

Definition 3. *Given a convex function $\psi : \mathbb{R}^p \to \mathbb{R}$, the proximal operator of ψ is the function* $\text{prox}_{\psi} : \mathbb{R}^p \to \mathbb{R}$ *defined by*

$$\text{prox}_{\psi}(x) = \arg\min_{z\in\mathbb{R}^p} \left\{ \psi(z) + \frac{1}{2}\|z - x\|^2 \right\} \tag{1.27}$$

When the regularization ψ is relatively simple, the associated proximal operator has a closed form solution, which is the case for ℓ_1 regularizer, the Elastic-net regularizer or Group-Lasso without overlap [Combettes and Pesquet, 2011]. Moreover, when ψ is the indicator function $\mathbb{1}_{\mathcal{C}}$, the proximal operator is simply the orthogonal projection onto the convex domain $\mathcal{C}$. Thus, by using the proximal operator, gradient descent and accelerated gradient descent algorithms can be extended to solve composite problems such as

$$\min_{x\in\mathbb{R}^p} \{ f(x) = f_0(x) + \psi(x) \},$$

where f_0 is differentiable and L-smooth, representing the smooth part of the function. Proximal gradient methods perform alternatively the gradient descent step and the proximal step, yielding the class of *iterative shrinkage-thresholding algorithms* (ISTA) [Daubechies et al., 2004], also called *forward-backward splitting method* [Gabay, 1983, Combettes and Pesquet, 2011, Raguet et al., 2013, Stella et al., 2017]. The accelerated variant is later proposed in [Beck and Teboulle, 2009, Tseng, 2008, Nesterov, 2013], yielding an acceleration in the sense of Nesterov.

ISTA with constant stepsize
Initialization: $x_0 \in \mathbb{R}^d$ and $\eta = \frac{1}{L}$. For $k \geq 0$, $$x_{k+1} = \text{prox}_{\eta\psi}\left(x_k - \eta\nabla f_0(x_k)\right).$$

Compared to their smooth counterpart, proximal algorithms perform an additional proximal step after each gradient descent step. This can be interpreted as a maximization-minimization scheme, where the smooth part f_0 is approximated by a quadratic upper bound, namely

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^p} \left\{ f_0(z) + \langle \nabla f_0(z), x - z \rangle + \frac{L}{2}\|x - z\|^2 + \psi(x) \right\}, \tag{1.28}$$

where z is the center of the quadratic approximation. When ψ is the indicator of a convex set, the proximal operator projects the gradient step back to the constrained set, yielding a generalization of the projected gradient method. In terms of convergence, these proximal variants enjoy the exact same rate as the corresponding smooth variants. In practice, a more aggressive strategy to determine the stepsize is proposed by Beck and Teboulle [2009]. It consists in applying the backtracking line search and check whether a sufficient decrease is achieved. This strategy removes the dependency of the Lipschitz parameter L, which makes the method more practical.

FISTA with constant stepsize

input $x_0 \in \mathbb{R}^d$ and $\alpha_0 \in (0, 1)$.

1: Set $y_0 = x_0$, $\eta = \frac{1}{L}$ and $q = \frac{\mu}{L}$.

2: At k-th iteration, $(k \geq 0)$

(a) Compute $x_{k+1} = \text{prox}_{\eta\psi}(y_k - \eta \nabla f_0(y_k))$.

(b) Compute $\alpha_{k+1} \in (0, 1)$ from equation

$$\alpha_{k+1}^2 = (1 - \alpha_{k+1})\alpha_k^2 + q\alpha_{k+1}.$$

(c) Compute y_k with

$$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k) \quad \text{with} \quad \beta_k = \frac{\alpha_k(1 - \alpha_k)}{\alpha_k^2 + \alpha_{k+1}}.$$

While proximal gradient methods have gained a lot of success, efficiently extending Quasi-Newton methods to the composite setting is still not completely well understood. There are several attempts to develop a proximal Quasi-Newton method [Byrd et al., 2015, Lee et al., 2012, Scheinberg and Tang, 2016, Yu et al., 2008], providing the recurrence

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^p} \left\{ f_0(x_k) + \langle \nabla f_0(x_k), x - x_k \rangle + \frac{1}{2}(x - x_k)^T B_k (x - x_k) + \psi(x) \right\}. \tag{1.29}$$

This is a direct extension of (1.28) by incorporating the Hessian approximation B_k. However, the subproblem (1.29) does not have a closed form solution in general since B_k is varying from iteration to iteration and it is usually non isotropic. Thus, an additional effort is required to solve the subproblem by applying an optimization algorithm, giving an inexact solution for the proximal operator. If the errors of the subproblem are small enough [Byrd et al., 2015], then the inexact proximal Newton method enjoys the same rate as in the smooth case in terms of the number of iterations. But the per-iteration cost is much higher. Moreover, the computational effort to solve the subproblem is hard to characterize since the condition number of B_k may be poor. In practice, the heuristic of applying several passes of the randomized coordinate descent is commonly used. Another drawback is that the storage of the entire matrix B_k seems inevitable, ruling out the limited memory variant.

Besides the proximal operator, another way to handle non differentiability is to introduce the notion of subgradients [Hiriart-Urruty and Lemaréchal, 1996]:

Definition 4. *Given a convex function $f : \mathbb{R}^p \to \mathbb{R}$, for any point $x \in \mathbb{R}^p$, we define*

$$\partial f(x) = \{g \in \mathbb{R}^p \,|\, \forall y \in \mathbb{R}^p, f(y) \geq f(x) + \langle g, y - x\rangle\}. \tag{1.30}$$

When f is differentiable at x, the set of the subgradients $\partial f(x)$ reduces to the singleton $\{\nabla f(x)\}$. Then, a natural generalization of the gradient method is to replace the gradient $\nabla f(x_k)$ by an element g_k in the subdifferential set $\partial f(x_k)$. However, the stepsize of the subgradient method needs to be decreasing in order to ensure convergence, giving a much slower convergence rate compared to the smooth case. The number of iterations required to reach an ε-solution is $O(\frac{1}{\varepsilon^2})$ when the problem is convex and is $O(\frac{1}{\varepsilon})$ when the problem is strongly convex [Bertsekas, 1999, Shor, 2012]. Moreover, when we are looking for a sparse solution, the subgradient method usually will not lead to coordinates equal to zero. In contrast, the proximal gradient methods do provide sparse solutions by taking into account the composite structure.

1.5 A survey of incremental algorithms

For modern machine learning problems, both the amount of data n and the dimension of features d can be very large, which excludes higher-order optimization methods. Under this setting, the gradient methods that we have presented so far are also computationally expensive, because evaluating the full gradient requires to pass through all the n training points. This motivates the development of incremental gradient methods which have a cost per-iteration independent of n.

Let us start with the case where the objective function is smooth, i.e.,

$$\min_{x \in \mathbb{R}^p} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}, \tag{1.31}$$

and n is large. Due to the decomposable structure of the objective, incremental gradient methods can operate on a single component f_i at each iteration instead of the entire sum in

the objective. This reduces the computational complexity per iteration by a factor n. If each iteration of the incremental method yields reasonable progress in the objective, the resulting incremental method will outperform its non-incremental counterpart. This point explains the success of incremental methods in a wide variety of applications.

One of the best known incremental method is the stochastic gradient descent method (SGD), which generates a sequence

$$x_{k+1} = x_k - \eta_k \nabla f_{i_k}(x_k), \tag{1.32}$$

where η_k is the stepsize and i_k is an index uniformly generated among 1 to n. Intuitively, we are performing an inexact gradient method at each iteration, where the full gradient $\nabla f(x_k)$ is approximated by the incremental gradient $\nabla f_{i_k}(x_k)$. The difference $\nabla f_{i_k}(x_k) - \nabla f(x_k)$ represents the noise of the approximation and its magnitude is controlled by the variance

$$\mathbb{E}_{i_k}[\|\nabla f_{i_k}(x_k) - \nabla f(x_k)\|^2] = \frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(x_k) - \nabla f(x_k)\|^2.$$

In general, this variance is non zero even at the minimum x^*, where the full gradient $\nabla f(x^*)$ vanishes but $\nabla f_i(x^*)$ may not. Thus assuming that at some iteration we attain the exact minimum $x_k = x^*$, we can move away from it at the next iteration, meaning that x^* is not a fixed point with respect to such dynamics. As a consequence, in order to ensure the convergence to the minimum, we need to decrease the stepsize η_k, which reduces the magnitude of error. It is shown that with the choice $\eta_k = O(1/k)$, the algorithm achieves a sublinear convergence rate. A proximal version of it is given in [Bertsekas, 2011] enjoying a similar rate. However, unlike full gradient methods for which linear convergence can be obtained, the stochastic gradient descent does not converge faster than $O(1/k)$ even for strongly convex functions. In practice, we also observe that it converges quickly in the first iterations but then slows down and attain some saturation due to the decreasing stepsize [Bottou, 2010].

A lot of recent work has been devoted to the study of faster incremental methods, including SAG [Schmidt et al., 2017], SAGA [Defazio et al., 2014a], Finito/MISO [Defazio et al., 2014b, Mairal, 2015], SDCA [Shalev-Shwartz and Zhang, 2012] and SVRG [Xiao and Zhang, 2014]. All of them converge linearly for strongly convex objectives, which significantly improves the performance of the stochastic gradient descent method. The main idea is to reduce the variance of the stochastic update such that the magnitude of the noise goes to zero while the iterates $(x_k)_{k\geq 0}$ converge to x^*. This will allow constant stepsizes, leading to a faster convergence.

Primal variance-reduction methods. The key to reduce the variance is to incorporate information regarding past iterates. For primal methods such as SAG, SAGA and SVRG, the update is given by:

$$x_{k+1} = x_k - \eta \left[\frac{\nabla f_{i_k}(x_k) - g_{i_k}}{\sigma} + \frac{1}{n}\sum_{i=1}^{n} g_i \right]. \tag{1.33}$$

where η is the stepsize, σ is a scaling parameter and g_i represents a past incremental gradient of the component f_i, for $i \in [1, n]$. The natural choice of σ is 1, in which case,

$$\mathbb{E}_{i_k}\left[\nabla f_{i_k}(x_k) - g_{i_k} + \frac{1}{n}\sum_{i=1}^{n} g_i\right] = \nabla f(x_k),$$

leading to an unbiased gradient update. This is the case for SVRG and SAGA. Although they enjoy a similar update expression, a fundamental difference arises in the way of updating g_i.

1. In SVRG, all the g_i's are updated simultaneously every m iterations, where m is a predefined parameter. More precisely, at initialization, all g_i's are initialized by $\nabla f_i(x_0)$, for all $i \in [1, n]$; then they are kept unchanged during m iterations; at the m-th iteration, they are simultaneously updated by the gradients at x_m namely $\nabla f_i(x_m)$. Then again unchanged in the next m iterations, and the next update will be performed at $2m$-th iteration, and so on.

2. In SAGA, the strategy is different. SAGA updates one g_i at each iteration. At k-th iteration, one index $i_k \in [1, n]$ is selected and x_{k+1} is obtained following (1.33). After that, g_{i_k} is updated by the latest gradient, namely $g_{i_k} = \nabla f_{i_k}(x_k)$. Then we move to the next iteration.

Intuitively, SVRG updates the past gradients with delay while SAGA updates them in an online fashion, which makes SAGA more aggressive but less stable. SAG updates the past gradient in an identical way as SAGA does, the only difference is that SAG chooses $\sigma = n$ instead of $\sigma = 1$. The biased gradient update provides further reduction of the variance, see [Defazio et al., 2014a] for more details.

The difference in terms of the update strategy leads to different requirements in terms of memory. For SVRG, the required memory is two vectors of dimension d, one for the iterate x_{km} and the other one for the full gradient $\nabla f(x_{km})$;[3] for SAGA, the required memory is n vectors of dimension d, one for each g_i. This makes a huge difference, since when $n \times d$ is large, the amount of memory required by SAGA becomes prohibitive. For the incremental methods we presented here, including the upcoming dual methods SDCA, Finito or MISO, SVRG is the only one enjoying such "memory-free" property. Fortunately, for problems involving linear models, it is possible to reduce the memory requirement of these methods from n vectors to n scalars.

Dual variance reduction methods. Now we consider the dual-based incremental algorithms, such as SDCA and Finito/MISO. In contrast to the primal algorithm which controls the primal gap $f(x_k) - f^*$, dual-based algorithms construct some lower bounds d_k of the objective function and ensure the convergence of the duality gap $f^* - d_k(x_k)$. In SDCA, the stochastic

[3] As a recompense, we need to additionally evaluate g_{i_k} at each iteration.

coordinate ascent algorithm is applied on the dual problem, which is obtained thanks to the Fenchel conjugate. Amazingly, while SDCA uses the concept of duality in behind, the update of the algorithm can be performed without evaluating the dual function, named as dual-free SDCA [Shalev-Shwartz, 2016]. The advantage of such an approach comes from the fact that the gradient of the Fenchel conjugate is not always easy to evaluate, a purely primal update makes the algorithm more practical. Closely related to SDCA, MISO substitutes the Fenchel conjugate by the quadratic lower bounds given by the strong convexity. Since the quadratic functions are more computationally friendly compared to the convex conjugate function, MISO provides systematically a quantity $d_k(x_k)$ lower bounding f^*.

Despite the difference in the choice of the lower bound function, SDCA and MISO enjoy a very similar update rule. Both of them keep in memory n auxiliary vectors $z_1, \dots, z_n$, one for each component f_i. At each iteration, an index $i_k \in [1, n]$ is randomly selected and the corresponding auxiliary vector is updated by:

$$\text{(SDCA \& MISO)} \quad z_{i_k} = (1-\delta) z_{i_k} + \delta \left(x_k - \frac{1}{\mu} \nabla f_{i_k}(x_k) \right).$$

Then, the next iterate is obtained by averaging over all of the auxiliary vectors:

$$x_{k+1} = \frac{1}{n} \sum_{i=1}^{n} z_i.$$

Each z_i carries the information of the i-th component and contributes equally to the iterate x_k, which reflects the averaging structure of f. The only difference between SDCA and MISO comes from the choice of the parameter δ: in SDCA $\delta = \frac{\mu n}{L+\mu n}$ and in MISO, $\delta = \min\{1, \frac{\mu n}{2(L-\mu)}\}$. It is interesting to see that different approaches converging to a similar algorithm even though the proof cannot be adapted from one to another. The benefit of dual-based methods, is that an upper bound on the function value gap $f(x_k) - f^*$ is provided by the duality gap $f(x_k) - d_k(x_k)$, which is easily computable and thus can be used to stop the algorithm. In general, the function value gap $f(x_k) - f^*$ is not available for primal algorithms simply because f^* is unknown. When the function is smooth, the norm of the gradient can be used as a substitution, but it will be more challenging for non smooth functions.

Another related direction is to develop incremental algorithms for the dual-averaging method [Nesterov, 2009]. The main idea is to move along the accumulated gradient, namely

$$x_{t+1} = x_t - \frac{\eta}{t} \sum_{k=1}^{t} \nabla f(x_k). \tag{1.34}$$

An incremental version of this has been provided by Xiao [2010] by replacing the full gradient $\nabla f(x_k)$ by the incremental gradient $\nabla f_{i_k}(x_k)$ in (1.34). The resulting algorithm enjoys a similar convergence rate as SGD, which is sublinear even for strongly convex objectives. Variance-reduced variants have been later studied by Murata and Suzuki [2016], mimicking the approach of SVRG.

So far, we have only considered smooth problems. In order to adapt to the composite setting, proximal variants are typically developed based on the original smooth incremental method [Xiao, 2010, Shalev-Shwartz and Zhang, 2012, Xiao and Zhang, 2014, Defazio et al., 2014a]. The first contribution of this manuscript is to introduce a proximal variant of Finito/MISO algorithm and provide the convergence analysis, which is the main focus of Chapter 2.

The comments about convergence and memory for the smooth versions still hold for the proximal versions: proximal SVRG is the only algorithm that requires an amount of $O(d)$ space in memory instead of $O(nd)$. In summary, the mentioned variance reduction incremental algorithms all enjoy linear convergence rates when the objective function is strongly convex. This outperforms classical SGD algorithms in terms of optimization error. The downside is that a non negligible amount of information needs to be stored in memory, except for SVRG. Nevertheless, the performance of SVRG is often slower in practice than the other memory-consuming methods. Thus, we have to make a choice regarding the trade-off between memory and speed.

Compared to full gradient methods, the per-iteration cost of incremental methods is n times smaller. In order to make a fair comparison, we count the computational complexity in the numbers of evaluations for a single gradient ∇f_i. Thus evaluating the full gradient requires to sum up the gradient of each component, yielding a complexity $O(n)$. The computational complexity for reaching an ε-solution is provided in the following table:

In the strongly convex setting, the complexity of full gradient methods (FG) depends multiplicatively on the number of components n and the condition number L_f/μ. In comparison, incremental methods like SAG/SAGA, Finito/MISO, SDCA and SVRG all enjoy a better complexity, where the dependence on the number of components and the condition number is additive. This improvement becomes significant when the condition number is large. In typical machine learning problems like ridge regression, the regularization parameter is in the order of $1/n$ giving a condition number in the order of n, in which case the full gradient methods lose a large factor n compared to incremental methods.

In contrast, when the problem is not strongly convex, there is no theoretical improvement in terms of complexity; SAG/SAGA enjoys the same complexity as full gradient methods. Moreover, several algorithms like Finito/MISO, SDCA and SVRG are not defined in the non strongly convex setting. Thus two natural questions arise: are these algorithms optimal and can we adapt them in the convex but non-strongly convex settings?

The first accelerated incremental method was developed by Shalev-Shwartz and Zhang [2016], where an accelerated version of SDCA has been proposed. The complexity of the resulting algorithm is given in Table 1.1, where the dependence on the condition number is improved by its squared root, yielding an acceleration in the sense of Nesterov. Compared to SDCA, the acceleration occurs when the condition number L/μ is larger than n, *e.g.*, for poorly

[4]The L we are using here is in fact the maximum of L_i, where L_i is the Lipschitz constant with respect to the i-th component f_i. An improvement of this dependency to the average Lipschitz constant can be achieved by applying a non uniform sampling strategy; we omit the discussion here for simplicity.

	Strongly convex: $\mu > 0$	Convex: $\mu = 0$
FG	$O\left(n\frac{L_f}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$	$O\left(n\frac{L_f}{\varepsilon}\right)$
SAG	$O\left(\left(n+\frac{L}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$ [4]	$O\left(n\frac{L}{\varepsilon}\right)$
SAGA		
Finito/MISO		not avail.
SDCA		
SVRG		
Acc-FG	$O\left(n\sqrt{\frac{L_f}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$	$O\left(n\sqrt{\frac{L_f}{\varepsilon}}\right)$
Acc-SDCA	$\tilde{O}\left(\left(n+\sqrt{\frac{nL}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$	not avail.
Lower bounds	$O\left(\left(n+\sqrt{\frac{nL}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$	$O\left(\sqrt{\frac{nL}{\varepsilon}}\right)$

Table 1.1 – Comparison of rates of convergence, respectively in the strongly-convex and non strongly-convex cases. The notation $\tilde{O}$ for acc-SDCA hides logarithmic terms in the condition number and n. We remark that there is a difference between the Lipschitz constant L_f used for full gradient method and the one L used in incremental methods. In full gradient method, the Lipschitz constant is with respect to the entire function f, whereas each component f_i needs to be L-smooth for incremental methods. Thus, we always have $L_f \leq L$, but L_f may be much smaller than L. Incremental methods are thus outperforming full gradient methods for the finite-sum problem we consider here *unless there is a big mismatch between L and L_f*.

conditioned problems. However, the provided analysis is restricted to strongly convex problems like the original SDCA. A more general scheme has been developed later by Frostig et al. [2015] for smooth strongly convex problems.

Whether or not incremental algorithms other than SDCA can be accelerated in the composite setting remained an open question at the beginning of this thesis. Moreover, it was also not clear whether an acceleration may be obtained for convex but not strongly convex problems. This brings us to one of the main contribution of this thesis. We present in Chapter 3 the first generic acceleration scheme called Catalyst [Lin et al., 2015], which provides a positive answer for both open questions. It can be applied to a large class of gradient-based algorithms, yielding the acceleration in the sense of Nesterov for both strongly convex and non-strongly convex objectives.

The lower bound for the class of incremental methods have been investigated later by Agarwal and Bottou [2015], Woodworth and Srebro [2016], Arjevani and Shamir [2016]. It is shown that accelerated SDCA [Shalev-Shwartz and Zhang, 2016] and Catalyst [Lin et al., 2015] are optimal

up to a logarithmic factor. A lot of other works have recently been developed towards obtaining optimal incremental methods, such as APCG [Lin et al., 2014], SDPC [Zhang and Xiao, 2015a], RPDG [Lan and Zhou, 2015], Point-SAGA [Defazio, 2016] and Katyusha [Allen-Zhu, 2016]. We remark that their techniques are algorithm-specific and cannot be directly generalized into a unified scheme. These methods enjoy the tight optimal convergence rate while our method loses a logarithmic factor. This may be the price to pay for providing a generic framework.

While first-order incremental algorithms have caught a lot of attention, incorporating curvature information in the large finite sum setting remains a challenging problem. The direct application of substituting the full gradient ∇f by an incremental gradient ∇f_i in the Quasi-Newton update is unfortunately not convergent. The reason is that such a substitution does not provide unbiased estimation of the Quasi-Newton update $B_k^{-1}\nabla f$ since there is a matrix inversion. Thus, it seems hopeless to incrementally update both the gradient and the Hessian estimate at the same time. An alternative for getting rid of this issue is to use a sub-sampled Hessian for estimating B_k. This gives the so called stochastic Quasi-Newton method (SQN) [Mokhtari and Ribeiro, 2015, Byrd et al., 2016, Gower et al., 2016, Moritz et al., 2016]. However, the construction of SQN highly depends on the smoothness of the objective function, which makes it difficult to generalize to the composite setting. Moreover, as mentioned in section 1.4, proximal Quasi-Newton methods suffer the drawback of requiring a non isotropic proximal operator at each iteration, where an exact solution is infeasible in practice.

Thus, a theoretically grounded variable metric algorithm in the composite setting is still missing in the literature. In Chapter 4, we introduce the first generic variable metric framework, QuickeNing, applicable to the composite finite sum structure. The algorithm can be interpreted as applying an inexact L-BFGS algorithm to the classical smoothing technique called infimum convolution or the Moreau-Yosida regularization, that we will present in Section 1.6. Global convergence analysis is provided followed by promising experimental results.

We should mention that 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 bundle methods [Chen and Fukushima, 1999, Fukushima and Qi, 1996, Mifflin, 1996], which use BFGS updates on the Moreau-Yosida smoothing of the objective and bundle methods to approximately solve the corresponding sub-problems. However, no global complexity was known, especially no complexity guarantee was given regarding the complexity of solving the subproblems. Therefore, the first contribution of QuickeNing lies in the global convergence analysis. Moreover, QuickeNing can be applied to incremental algorithms. The resulting algorithm successfully incorporate the curvature information with the incremental algorithms as expected.

Up to now, we have only focused on the minimization of convex problems. However, many interesting machine learning applications involves non-convex problems in extremely large-scale

settings, such as neural networks. Thus, efficiently extending incremental algorithms to non-convex problems has high interest.

A variant of SVRG for non-convex problems has been recently introduced and analyzed [Reddi et al., 2016b]. One challenge we tackle in Chapter 5 is to extend all the previous incremental methods designed for convex objectives to work in a non-convex context. As we will see, we propose a variant of Catalyst that does not require any knowledge about the convexity of the function to operate.

1.6 The Moreau-Yosida regularization

In this section, we introduce a fundamental tool from convex analysis called the Moreau-Yosida envelope [Moreau, 1962, Yosida, 1980], which plays a key role for understanding generic algorithms such as Catalyst and QuickeNing. This tool can be seen as a smoothing technique, that can turn any convex lower semicontinuous function f into a smooth function with Lipschitz gradients. More precisely, it consists of the infimal convolution of f with a quadratic penalty:

$$F(x) \triangleq \min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|z - x\|^2 \right\}, \tag{1.35}$$

where κ is a positive regularization parameter. Recalling that the proximal operator is then defined as the unique minimizer of the problem—that is,

$$p(x) \triangleq \operatorname{prox}_{f/\kappa}(x) = \arg\min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|z - x\|^2 \right\}. \tag{1.36}$$

Unlike the proximal operator of the simple regularizer ψ defined in (1.27), $p(x)$ does not admit a closed-form solution in general. Therefore, computing it requires to solve a sub-problem with high accuracy thanks to some iterative algorithm.

Basic Properties of the Moreau-Yosida Envelope. The smoothing effect of the Moreau-Yosida regularization can be characterized by the next proposition [see Lemaréchal and Sagastizábal, 1997a, for elementary proofs].

Proposition 1 (Regularization properties of the Moreau-Yosida Envelope). *Given a convex continuous function f and a regularization parameter $\kappa > 0$, we consider the Moreau-Yosida envelope F defined in (1.35). Then,*

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

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

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

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

$$\nabla F(x) = \kappa(x - p(x)). \tag{1.37}$$

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

3. *If f is μ-strongly convex, then F is μ_F-strongly convex with $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$.*

Interestingly, F inherits from the convexity of f and more importantly it is always L-smooth. Moreover, when f is μ-strongly convex, the condition number of F is given by

$$Q_F = \frac{L_F}{\mu_F} = \frac{\mu + \kappa}{\mu},$$

which can be arbitrarily well conditioned by choosing a small value of κ. Naturally, since both functions admit the same solutions, a naive approach for minimizing a non-smooth function f is to apply a smooth optimization algorithm on F. Different variants of proximal point algorithm are obtained by respectively applying the gradient descent method, the Nesterov's accelerated gradient method and Quasi-Newton methods on the Moreau-Yosida envelope F.

The proximal point algorithm. Let us consider gradient descent steps on F:

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

Noticing from (1.37), we have $\nabla F(x_k) = \kappa(x_k - p(x_k))$ and $L_F = \kappa$. Thus,

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

which is exactly the proximal point algorithm [Martinet, 1970, Rockafellar, 1976].

Accelerated proximal point algorithm. We now consider applying Nesterov's accelerated gradient method on F:

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

where β_{k+1} is Nesterov's extrapolation parameter [Nesterov, 2004]. Again from (1.37), the update can be rewritten as

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

which is known as the accelerated proximal point algorithm of Güler [1992].

Variable metric proximal point algorithm. Let us now apply Quasi-Newton methods on F:

$$x_{k+1} = x_k - H_k \nabla F(x_k) = x_k - \kappa H_k(x_k - p(x_k)),$$

where the inverse Hessian matrix H_k is constructed using previous gradients of F. This is known as the variable metric proximal point algorithm [Burke and Qian, 1999].

As a consequence, the generated iterates $(x_k)_{k\geq 0}$ converge to the minimum of F, which is also the minimum of f. Furthermore, the convergence rate of the above algorithms can be directly derived from the convergence of the underlying smooth optimization methods. Since the condition number of F can be arbitrarily good, the convergence of them can be arbitrarily fast with a small enough regularization parameter κ.

While these algorithms are conceptually elegant, they suffer from a major drawback in practice, which is the exact evaluation of the proximal operator $p(x)$ at each iteration. Unless a closed form is available, which is almost never the case, an iterative algorithm is required to approximate the proximal operator. This raises two essential issues: first, the approximate proximal operator encounters inexactness in the gradient evaluation, the magnitude of the inexactness should be carefully addressed in order to ensure the convergence of the algorithm; second, an additional computational effort is required by applying an iterative algorithm to approximate the proximal operator, the resulting computational cost needs to be controlled and included in the convergence analysis.

The inexactness criteria have been intensively investigated. Different criteria have been developed in the context of the proximal point algorithm, starting from the pioneer work of Rockafellar [1976], followed by Auslender [1987], Correa and Lemaréchal [1993], Solodov and Svaiter [2001], Fuentes et al. [2012]. While the proximal point algorithm has caught a lot of attention, very few works have focused on its accelerated variant. The first accelerated proximal point algorithm with inexact gradients was proposed by Güler [1992]. Then, Salzo and Villa [2012] developed a more rigorous convergence analysis, generalized later by Devolder et al. [2014]. However, the considered inexactness criteria for the accelerated variants are unpractical, leaving some space for improvement. As a direct extension, the strategy of combining Quasi-Newton methods and the Moreau-Yosida regularization has been applied to develop variable metric bundle methods [Fukushima and Qi, 1996, Mifflin, 1996, Lemaréchal and Sagastizábal, 1997b, Chen and Fukushima, 1999].

In the meantime, the computational complexity for approximating the proximal operator is not investigated. This is the reason why the global complexity of proximal point algorithms is not well understood. To the best of our knowledge, the first attempt in this direction is provided in the accelerated SDCA algorithm [Shalev-Shwartz and Zhang, 2016], which is an instance of inexact accelerated proximal point algorithm, even though this was not explicitly stated in the original paper. Their analysis only holds in the strongly convex setting and it was originally designed for the stochastic dual coordinate ascent method. Inspired by this, we extend their idea into generic acceleration schemes, providing global convergence analysis for different variants of the proximal point algorithms.

1.7 Contributions of this thesis

We outline here the main contributions of this thesis following the organization of the manuscript.

- In Chapter 2, we present Prox-MISO, the proximal version of the incremental algorithm Finito/MISO [Defazio et al., 2014b, Mairal, 2015]. The resulting algorithm extends Finito/MISO to the composite problems (1.26). Moreover, it removes the big data condition $n \geq 2\frac{L}{\mu}$ originally required in Finito/MISO. We show that the algorithm is linearly convergent when the problem is strongly convex and that the number of iterations required to achieve an ε-solution is given by

$$O\left(\left(n + \frac{L}{\mu}\right) \log\left(\frac{1}{\varepsilon}\right)\right). \tag{1.38}$$

  Since Prox-MISO builds a simple lower bound of the objective function, a practical optimality certificate is systematically obtained, which is a benefit compared to primal algorithms.

- In Chapter 3, we introduce a generic acceleration scheme that applies to a large class of algorithms. We call our approach a "catalyst" by analogy with substances that increase chemical reaction rates. Catalyst takes an optimization method $\mathcal{M}$ as input and applies it to solve a series of subproblems in order to achieve acceleration. A method $\mathcal{M}$ may be accelerated if it enjoys linear convergence rate for strongly convex problems. This is the case for full gradient methods [Beck and Teboulle, 2009, Nesterov, 2013] and block coordinate descent methods [Nesterov, 2012b, Richtárik and Takáč, 2014], which already have well-known accelerated variants. More importantly, it also applies to incremental algorithms such as SAG [Schmidt et al., 2017], SAGA [Defazio et al., 2014a], Finito/MISO [Defazio et al., 2014b, Mairal, 2015], SVRG [Xiao and Zhang, 2014] and SDCA [Shalev-Shwartz and Zhang, 2012]. Whether or not these methods could be accelerated was an important open question. It was only known to be the case for dual coordinate ascent approaches such as SDCA for strongly convex objectives [Shalev-Shwartz and Zhang, 2016].

  Our work provides a universal positive answer regardless of the strong convexity of the objective. Let us illustrate the phenomena of acceleration with the example of the Prox-MISO algorithm.

  1. When the problem is strongly convex, Catalyst applying to Prox-MISO yields the complexity

$$\tilde{O}\left(\left(n + \sqrt{n\frac{L}{\mu}}\right) \log\left(\frac{1}{\varepsilon}\right)\right), \tag{1.39}$$

     where $\tilde{O}$ hides logarithmic constant in terms of n and the condition number. The obtained complexity outperforms the one of (1.38) when the problem is ill-conditioned.

2. When the problem is convex, Catalyst applying to Prox-MISO yields the complexity

$$\tilde{O}\left(\sqrt{\frac{nL}{\varepsilon}}\right), \tag{1.40}$$

where $\tilde{O}$ hides logarithmic constant in terms of n, the condition number and $\frac{1}{\varepsilon}$. We should mention that the Prox-MISO algorithm is not defined for non-strongly convex objectives, but it is theoretically grounded to apply it inside Catalyst. Thus, our work automatically provides support for non-strongly convex objectives even though the input method is not defined under such setting. This is a side achievement besides the acceleration.

The resulting complexity in (1.39) and (1.40) match the minimax lower bounds later developed by Woodworth and Srebro [2016], Arjevani and Shamir [2016] up to a logarithmic factor. More generally, for all the incremental algorithms mentioned before, applying Catalyst provides both acceleration and support for non-strongly convex objectives with a similar convergence rate. Experiments are conducted to show that the acceleration is useful in practice, especially for ill-conditioned problems where we observe significant improvement.

- Chapter 4 is dedicated to another generic acceleration approach called QuickeNing, which uses the Quasi-Newton principles to accelerate gradient-based optimization methods. To the best of our knowledge, QuickeNing is the first Quasi-Newton type algorithm compatible with both the composite objectives and the finite sum setting.

  The algorithm admits a simple interpretation: it may be seen as applying the L-BFGS algorithm with inexact gradients to the Moreau-Yosida regularization of the objective. Given an optimization method $\mathcal{M}$, we apply $\mathcal{M}$ to solve subproblems which correspond to approximating gradients of the Moreau-Yosida regularization. As a result, our approach is generic, and can be applied to the same class of functions as Catalyst. A particular warm start strategy is proposed in order to avoid the line search step in the determination of the stepsize, which makes the algorithm practical.

  The global complexity of the algorithm is given, showing that a linear convergence rate can be obtained when the problem is strongly convex. This is the first global complexity result of the variable metric proximal point algorithm involving the complexity for solving the subproblems. More importantly, the convergence analysis provides us an explicit way to set up parameters, which are keys to obtain fast convergence in practice.

  We remark that QuickeNing algorithm does not provide any theoretical acceleration in the worst case complexity analysis. This comes from the fact that the theoretical guarantee of L-BFGS algorithm is not better than the gradient descent method. However, we provide extensive experiments showing that QuickeNing gives significant improvement over competing methods in large-scale machine learning problems.

- In Chapter 5, we extend Catalyst algorithm to the nonconvex setting. This is a joint work with Courtney Paquette and Dmitriy Drusvyatskiy from University of Washington. The strength of the approach lies in the ability of automatically adapting the convexity, meaning that we can run the algorithm without specifying that the objective is convex or not. A particular step called Auto-adapt is designed in the algorithm to exploit the local convexity. When the objective is convex, the proposed approach enjoys the same convergence rate as the original Catalyst algorithm, leading to an acceleration. When the objective is nonconvex, it achieves the best known convergence rate to stationary points for first-order methods. As a consequence, it automatically transforms a convex optimization method to adapt to the nonconvex setting. We present promising experimental results when applied it to problems of sparse matrix factorization and neural networks.

Chapter 2

Proximal minimization by incremental surrogate optimization (MISO)

Chapter abstract:
The approach MISO, proposed by Mairal [2015], is an incremental technique for minimizing unconstrained finite sum problems. The original algorithm is limited to smooth strongly convex problems and requires a big data condition such that the number of components n is larger than the condition number L/μ. In this chapter, we allow the MISO approach to deal with composite problems. Moreover, we remove the big data assumption with a new proof technique. The resulting algorithm provides systematically a lower bound of the optimum which is simple to compute, leading to a practical optimality certificate.

The material of this chapter is part of the following publication:

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

2.1 Introduction

Incremental algorithms have recently caught a lot of attention due to their ability of solving large-scale machine learning problems. Just like gradient descent methods, incremental algorithms are first studied in the smooth convex setting and extended later to composite problems. For instance, Proximal-SDCA [Shalev-Shwartz and Zhang, 2012] and Proximal-SVRG [Xiao and Zhang, 2014] are direct proximal extensions of SDCA [Shalev-Shwartz and Zhang, 2013] and SVRG [Johnson and Zhang, 2013]; SAGA [Defazio et al., 2014a] is an implicit proximal variant of SAG [Schmidt et al., 2017]. The smooth variants usually enjoy simpler expressions

and easier convergence analysis compared to the proximal variants. For this reason, the smooth setting is often considered in the first place.

Finito/MISO is another incremental approach, proposed simultaneously by Defazio et al. [2014b] and Mairal [2015] for solving smooth unconstrained μ-strongly convex problems. We will call it MISO in the following discussion, which refers to Minimization by Incremental Surrogate Optimization. The concept of surrogate is about to replace the original objective f by a simpler approximation h and minimize the approximation h instead of f. A typical example is the Lipschitz gradient surrogate. Given a point y, we define a majorant surrogate of f near the point y by

$$h_y : x \to f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|y - x\|^2.$$

Minimizing h_y gives the classical gradient descent update $y - \frac{1}{L}\nabla f(y)$. Thus, the gradient descent method can be viewed as sequentially minimizing the Lipchitz gradient surrogate near the previous iterate. In the same way, proximal gradient methods can be obtained from the proximal gradient surrogate which is the sum of the non-smooth regularization and the Lipschitz gradient surrogate. Other well known optimization methods including accelerated gradient methods, block coordinate descent method, Franck-Wolfe method can also be interpreted as the minimization of surrogate functions [Mairal, 2013].

2.2 The original MISO algorithm

Consider the following smooth unconstrained convex minimization problem

$$\min_{x \in \mathbb{R}^p} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}, \tag{2.1}$$

where each f_i is differentiable with L-Lipschitz continuous derivatives and are μ-strongly convex.

The idea of MISO[1] is to construct a list of quadratic surrogate functions $(d_i)_{i \in [1,n]}$, where d_i represents the "dual" of the component f_i. At each iteration, one of the surrogates is updated, using information of the latest iterate. More precisely, at k-th iteration, an index i_k is randomly picked up among $[1, n]$ and we perform the following update given the last iterate x_{k-1}:

$$d_i^k(x) = \begin{cases} f_i(x_{k-1}) + \langle \nabla f_i(x_{k-1}), x - x_{k-1} \rangle + \frac{\mu}{2}\|x - x_{k-1}\|^2 & \text{if} \quad i = i_k \\ d_i^{k-1}(x) & \text{otherwise} \end{cases},$$

where d_i^k denotes the i-th surrogate at the k-th iteration. Then, the iterate is updated as the minimum of the entire surrogate,

$$x_k = \arg\min_{x \in \mathbb{R}^p} \left\{ \bar{d}_k(x) = \frac{1}{n} \sum_{i=1}^{n} d_i^k(x) \right\}. \tag{2.2}$$

[1]Note that even though we call this algorithm MISO (or Finito), it was called MISOμ in [Mairal, 2015], whereas "MISO" was originally referring to an incremental majorization-minimization procedure that uses upper bounds of the functions f_i instead of lower bounds.

The subproblem (2.2) is nothing but a minimization of a quadratic function which can be easily solved. Let us denote z_i^k as the minimum point of d_i^k, namely the expression

$$d_i^k(x) = c_i^k + \frac{\mu}{2}\|x - z_i^k\|^2,$$

holds for some constant c_i^k. Then the update rule of z_i^k is given by

$$z_i^k = \begin{cases} x_{k-1} - \frac{1}{\mu}\nabla f_i(x_{k-1}) & \text{if} \quad i = i_k \\ z_i^{k-1} & \text{otherwise} \end{cases},$$

Consequently, the iterate x_k is simply the average of the centers z_i^k:

$$x_k = \frac{1}{n}\sum_{i=1}^n z_i^k = x_{k-1} + \frac{1}{n}\left(z_{i_k}^k - z_{i_k}^{k-1}\right).$$

Interestingly, since $\bar{d}_k$ is a lower-bound of f we have $\bar{d}_k(x_k) \leq f^*$. Thus, the quantity $f(x_k) - \bar{d}_k(x_k)$ can be used as an optimality certificate that upper-bounds $f(x_k) - f^*$. Furthermore, this certificate was shown to converge to zero with a rate similar to SAG/SDCA/SVRG/SAGA under the condition $n \geq 2L/\mu$ [Mairal, 2015]:

$$\mathbb{E}[f(x_k) - \bar{d}_k(x_k)] \leq C(1 - \tau_{\text{MISO}})^k (f(x_0) - \bar{d}_0(x_0)).$$

with $\tau_{\text{MISO}} = 1/(3n)$ (also refined in $1/(2n)$ in [Defazio et al., 2014b]).

However, we need to keep in memory the list of n centers z_i^k in order to iteratively update the x_k. This may be expensive if the dimension of the feature is large. It is worth remarking that in many machine learning problems, each function $f_i(x)$ has the specific form $f_i(x) = l_i(\langle x, w_i\rangle) + \frac{\mu}{2}\|x\|^2$. In such cases, the vectors z_i^k can be obtained by storing only $O(n)$ scalars, this remark is still valid for other incremental methods like SAG/SAGA/SDCA.

In the next section, we extend MISO to the composite optimization problem and remove the big data condition $n \geq 2L/\mu$.

2.3 The proximal MISO algorithm

We now consider the composite optimization problem below,

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

where the functions f_i are differentiable with L-Lipschitz derivatives and μ-strongly convex. As in typical composite optimization problems, ψ is convex but not necessarily differentiable. We assume that the proximal operator of ψ can be computed easily.

Taking the similar approach as the original MISO algorithm, we construct lower bounds of f, by adding the regularization penalty ψ,

$$x_k = \arg\min_{x \in \mathbb{R}^p} \left\{ D_k(x) = \frac{1}{n} \sum_{i=1}^{n} d_i^k(x) + \psi(x) \right\}. \tag{2.3}$$

In order to remove the big data condition $n \geq 2L/\mu$, we introduce a "shrinking" parameter δ in $(0, 1)$. Let us denote l_i^k as the latest lower bound with respect to the component f_i,

$$l_i^k(x) = f_i(x_{k-1}) + \langle \nabla f_i(x_{k-1}), x - x_{k-1} \rangle + \frac{\mu}{2} \|x - x_{k-1}\|^2.$$

Then we perform the update of the surrogates d_i^k by following

$$d_i^k(x) = \begin{cases} (1-\delta) d_i^{k-1}(x) + \delta l_i^k(x) & \text{if} \quad i = i_k, \\ d_i^{k-1}(x) & \text{otherwise.} \end{cases} \tag{2.4}$$

This immediately implies the following algorithm.

Algorithm 1 Prox-MISO: MISO algorithm with proximal support.

input $(z_i^0)_{i=1,\ldots,n}$ such that (A1) holds; N (number of iterations);

1: initialize $\bar{z}_0 = \frac{1}{n} \sum_{i=1}^n z_i^0$ and $x_0 = \text{prox}_{\psi/\mu}[\bar{z}_0]$;

2: define $\delta = \min(1, \frac{\mu n}{2(L-\mu)})$;

3: **for** $k = 1, \ldots, N$ **do**

4: randomly pick up an index i_k in $[1, n]$ and update

$$\begin{aligned} z_i^k &= \begin{cases} (1-\delta) z_i^{k-1} + \delta \left(x_{k-1} - \frac{1}{\mu} \nabla f_i(x_{k-1}) \right) & \text{if} \quad i = i_k \\ z_i^{k-1} & \text{otherwise} \end{cases} \\ \bar{z}_k &= \bar{z}_{k-1} + \frac{1}{n} \left(z_{i_k}^k - z_{i_k}^{k-1} \right) = \frac{1}{n} \sum_{i=1}^n z_i^k \\ x_k &= \text{prox}_{\psi/\mu}[\bar{z}_k]. \end{aligned} \tag{2.5}$$

5: **end for**

output x_N (final estimate).

To illustrate the difference from the algorithm presented in the last section, we consider a pathological case when the sample size $n = 1$. If we apply the original MISO algorithm, it gives the gradient descent method with a large stepsize $1/\mu$, which has no guarantee of convergence. If we apply the new update, we take a less aggressive step with the stepsize shrunk by δ, which reduces the variance of the update. This is the key for removing the big data condition. Then, we remark that under the large sample size condition $n \geq 2L/\mu$ and $\psi = 0$, we have $\delta = 1$ and the update of the quantities z_i^k in (2.5) is identical as the original MISO algorithm. More generally, the choice of the parameter δ is driven from the convergence analysis that we will discuss later.

Relationship with the variance reduction approch. When the problem is smooth, i.e. $\psi = 0$, the Prox-MISO algorithm boils down to the following update:

$$x_k = x_{k-1} + \frac{\delta}{n}\left(x_{k-1} - \frac{1}{\mu}\nabla f_{i_k}(x_{k-1}) - z_{i_k}^{k-1}\right),$$

where i_k is the index randomly selected at k-th iteration. Remarking that

$$\mathbb{E}_{i_k}\left[x_{k-1} - \frac{1}{\mu}\nabla f_{i_k}(x_{k-1}) - z_{i_k}^{k-1}\right] = -\frac{1}{\mu}\nabla f(x_{k-1}),$$

the update of MISO is thus an unbiased gradient update. It enjoys a similar variance reduction interpretation as the primal incremental methods discussed in Section 1.5.

Relationship with Proximal SDCA [Shalev-Shwartz and Zhang, 2012]. The algorithm Prox-MISO is almost identical to variant 5 of proximal SDCA [Shalev-Shwartz and Zhang, 2012], which performs the same updates with $\delta = \mu n/(L + \mu n)$ instead of $\delta = \min(1, \frac{\mu n}{2(L-\mu)})$. It is however not clear that Prox-MISO actually performs dual ascent steps in the sense of SDCA since the proof of convergence of SDCA cannot be directly modified to use the stepsize of proximal MISO and furthermore, the convergence proof of Prox-MISO does not use the concept of duality. Another difference lies in the optimality certificate of the algorithms. Whereas Proximal-SDCA provides a certificate in terms of linear convergence of a duality gap based on Fenchel conjugate, Proximal-MISO ensures linear convergence of a gap that relies on strong convexity but not on the Fenchel conjugate (at least explicitly).

Relationship with Dual Averaging methods. The Prox-MISO algorithm is closely related to the dual averaging methods when the function f_i enjoys a particular form $f_i(x) = \tilde{f}_i(x) + \frac{\mu}{2}\|x\|^2$. In this case, the update of z_i^k is given by,

$$z_i^k = (1-\delta)z_i^{k-1} - \frac{\delta}{\mu}\nabla\tilde{f}_i(x_{k-1}).$$

Thus, the quantities z_i^k are simply non uniform average of the past gradients which plays a similar rule as the dual averaging term in the update of RDA algorithm [Xiao, 2010].

Initialization of surrogates. We now take a particular focus on the initialization of the surrogates that we have omitted to discuss so far. The algorithm needs to be initialized with some quadratic lower bounds for the functions f_i:

$$f_i(x) \geqslant d_i^0(x) \triangleq \frac{\mu}{2}\|x - z_i^0\|^2 + c_i^0, \tag{A1}$$

which are guaranteed to exist due to the μ-strong convexity of f_i. For typical machine learning applications, such initialization is easy. For example, logistic regression with ℓ_2-regularization

satisfies (A1) with $z_i^0 = 0$ and $c_i^0 = 0$. Then, the Prox-MISO scheme is given in Algorithm 1. Note that if no simple initialization is available, we may consider any initial estimate $\bar{z}_0$ in $\mathbb{R}^p$ and define $z_i^0 = \bar{z}_0 - (1/\mu)\nabla f_i(\bar{z}_0)$, which requires performing one pass over the data.

Optimality certificate and stopping criterion. Similar to the original MISO algorithm, Proximal MISO maintains a list (d_i^k) of lower bounds of the functions f_i. As a consequence, the following function is a lower bound of the objective f:

$$D_k(x) = \frac{1}{n}\sum_{i=1}^{n} d_i^k(x) + \psi(x), \tag{2.6}$$

and the update (2.5) can be shown to exactly minimize D_k. As a lower bound of f, we have that $D_k(x_k) \leq f^*$ and thus

$$f(x_k) - f^* \leq f(x_k) - D_k(x_k).$$

The quantity $f(x_k) - D_k(x_k)$ can then be interpreted as an optimality gap, and the analysis below will show that it converges linearly to zero.

Convergence analysis. We show that the certificate $f(x_k) - D_k(x_k)$ enjoys a linear convergence rate:

Theorem 6 (Convergence of Prox-MISO). *Let $(x_k)_{k\geq 0}$ be obtained by Prox-MISO, then*

$$\mathbb{E}[f(x_k) - D_k(x_k)] \leqslant \frac{1}{\tau}(1-\tau)^k \left(f^* - D_0(x_0)\right) \quad \textit{with} \quad \tau \geqslant \min\left\{\frac{\mu}{4L}, \frac{1}{2n}\right\}.$$

Furthermore, the convergence of $f(x_k) - f^*$ can be directly derived from it,

$$\mathbb{E}[f(x_k)] - f^* \leqslant \frac{1}{\tau}(1-\tau)^{k+1} \left(f(x_0) - D_0(x_0)\right). \tag{2.7}$$

Before we prove this theorem, we note that this rate is slightly better than the one proven in MISO [Mairal, 2015], which converges as $(1 - \frac{1}{3n})^k$. The main idea of the proof is to show that the lower bound $D_k(x_k)$ converges to f^* in a linear rate, acting as a dual method. In order to build connections between $D_k(x_k)$ and $D_{k-1}(x_{k-1})$, we introduce an intermediate term $D_k(x_{k-1})$ and control respectively the difference $D_k(x_{k-1}) - D_k(x_k)$ and $D_k(x_{k-1}) - D_{k-1}(x_{k-1})$. We start by recalling a classical lemma that provides useful inequalities. Its proof may be found in [Nesterov, 2004].

Lemma 1 (Classical Quadratic Upper and Lower Bounds). *For any function $h : \mathbb{R}^p \to \mathbb{R}$ which is μ-strongly convex and differentiable with L-Lipschitz derivatives, we have for all x, y in $\mathbb{R}^p$,*

$$\frac{\mu}{2}\|x-y\|^2 \leq h(x) - h(y) + \langle \nabla h(y), x - y\rangle \leq \frac{L}{2}\|x-y\|^2.$$

Then, we proceed with a sequence of technical lemmas

Lemma 2 (**Lower Bound on** $D_k - D_{k-1}$)**.** *For all* $k \geq 1$ *and* x *in* $\mathbb{R}^p$,

$$D_k(x) \geqslant D_{k-1}(x) - \frac{\delta(L-\mu)}{2n}\|x - x_{k-1}\|^2, \quad \forall x \in \mathbb{R}^p. \tag{2.8}$$

Proof. For all $i \in [1, n]$, f_i satisfies the assumptions of Lemma 1. Therefore, for all $k \geq 0$ and $i = i_k$, the following inequality holds for all x in $\mathbb{R}^p$,

$$\begin{aligned} d_i^k(x) &= (1-\delta)d_i^{k-1}(x) + \delta[f_i(x_{k-1}) + \langle \nabla f_i(x_{k-1}), x - x_{k-1}\rangle + \frac{\mu}{2}\|x - x_{k-1}\|^2] \\ &\geqslant (1-\delta)d_i^{k-1}(x) + \delta f_i(x) - \frac{\delta(L-\mu)}{2}\|x - x_{k-1}\|^2 \\ &\geqslant d_i^{k-1}(x) - \frac{\delta(L-\mu)}{2}\|x - x_{k-1}\|^2, \end{aligned}$$

where the first inequality uses Lemma 1, and the last one uses the inequality $f_i \geq d_i^{k-1}$. From this inequality, we can obtain (2.8) by simply using $D_k(x) = D_{k-1}(x) + \frac{1}{n}\left(d_{i_k}^k(x) - d_{i_k}^{k-1}(x)\right)$. □

Lemma 3 (**Relation between** D_k **and** D_{k-1})**.** *For all* $k \geq 0$*, for all* x *and* y *in* $\mathbb{R}^p$,

$$D_k(x) - D_k(y) = D_{k-1}(x) - D_{k-1}(y) - \mu\langle \bar{z}_k - \bar{z}_{k-1}, x - y\rangle. \tag{2.9}$$

Proof. This is straightforward from the fact that D_k is the sum of ψ and a quadratic function centered at $\bar{z}_k$, namely

$$D_k(x) = C_k + \frac{\mu}{2}\|x - \bar{z}_k\|^2 + \psi(x).$$

with some constant $C_k \in \mathbb{R}$. □

Lemma 4 (**Lower Bound on** $D_k(x_{k-1})$)**.** *For all* $k \geq 1$*, we have*

$$D_k(x_{k-1}) \geq D_{k-1}(x_{k-1}) + \frac{n\mu^2}{2\delta(L-\mu)}\|\bar{z}_k - \bar{z}_{k-1}\|^2. \tag{2.10}$$

Proof. Take $x = x_{k-1}$ and $y = x_{k-1} + \frac{n\mu}{\delta(L-\mu)}(\bar{z}_k - \bar{z}_{k-1})$ in (2.9) gives,

$$\begin{aligned} D_k(x_{k-1}) - D_{k-1}(x_{k-1}) &= D_k(y) - D_{k-1}(y) + \mu\langle \bar{z}_k - \bar{z}_{k-1}, y - x_{k-1}\rangle \\ \text{by (2.8)} &\geqslant -\frac{\delta(L-\mu)}{2n}\|y - x_{k-1}\|^2 + \mu\langle \bar{z}_k - \bar{z}_{k-1}, y - x_{k-1}\rangle \\ &= \frac{n\mu^2}{2\delta(L-\mu)}\|\bar{z}_k - \bar{z}_{k-1}\|^2. \end{aligned}$$

□

Lemma 5 (**Upper Bound on** $D_k(x_{k-1})$)**.** *For any* $k \geqslant 1$,

$$D_k(x_{k-1}) \leq D_k(x_k) + \frac{\mu}{2}\|\bar{z}_k - \bar{z}_{k-1}\|^2. \tag{2.11}$$

Proof. Apply Lemma 3 with $x = x_{k-1}$ and $y = x_k$ yields

$$D_k(x_{k-1}) - D_k(x_k) = D_{k-1}(x_{k-1}) - D_{k-1}(x_k) - \mu\langle \bar{z}_k - \bar{z}_{k-1}, x_{k-1} - x_k\rangle.$$

Since x_{k-1} is the minimum of D_{k-1} which is μ-strongly convex. Thus,

$$D_{k-1}(x_{k-1}) + \frac{\mu}{2}\|x_k - x_{k-1}\|^2 \leqslant D_{k-1}(x_k).$$

As a consequence,

$$D_k(x_{k-1}) - D_k(x_k) \leqslant -\frac{\mu}{2}\|x_k - x_{k-1}\|^2 - \mu\langle \bar{z}_k - \bar{z}_{k-1}, x_{k-1} - x_k\rangle \leqslant \frac{\mu}{2}\|\bar{z}_k - \bar{z}_{k-1}\|^2.$$

□

We are now in shape to prove the main convergence result.

Proof of Theorem 6. From Lemma 4 and Lemma 5, we have

$$\begin{aligned} D_k(x_{k-1}) - D_{k-1}(x_{k-1}) &\geq \frac{n\mu^2}{2\delta(L-\mu)}\|\bar{z}_k - \bar{z}_{k-1}\|^2 \\ &\geq \frac{n\mu}{\delta(L-\mu)}\left[D_k(x_{k-1}) - D_k(x_k)\right]. \end{aligned} \tag{2.12}$$

Recall that by construction,

$$D_k(x_{k-1}) = D_{k-1}(x_{k-1}) + \frac{\delta}{n}(f_{i_k}(x_{k-1}) - d_{i_k}^{k-1}(x_{k-1})).$$

Take expectation on i_k gives,

$$\mathbb{E}[D_k(x_{k-1})] = \left(1 - \frac{\delta}{n}\right)\mathbb{E}[D_{k-1}(x_{k-1})] + \frac{\delta}{n}\mathbb{E}[f(x_{k-1})]. \tag{2.13}$$

Apply (2.13) to (2.12) and rearrange the expression gives

$$\tau\mathbb{E}[f(x_{k-1})] - \mathbb{E}[D_k(x_k)] \leq -(1-\tau)\mathbb{E}[D_{k-1}(x_{k-1})].$$

where

$$\tau = \left(1 - \frac{\delta(L-\mu)}{n\mu}\right)\frac{\delta}{n}, \tag{2.14}$$

Interpolating with f^* yields

$$\tau\left(\mathbb{E}[f(x_{k-1})]-f^*\right)+\left(f^*-\mathbb{E}[D_k(x_k)]\right)\leqslant(1-\tau)\left(f^*-\mathbb{E}[D_{k-1}(x_{k-1})]\right). \tag{2.15}$$

On one hand, since $f(x_{k-1})\geqslant f^*$, we have

$$\begin{aligned} f^*-\mathbb{E}[D_k(x_k)] &\leq (1-\tau)\left(f^*-\mathbb{E}[D_{k-1}(x_{k-1})]\right) \\ &\leq \cdots \leq (1-\tau)^k\left(f^*-D_0(x_0)\right). \end{aligned} \tag{2.16}$$

On the other hand, since $f^*\geqslant D_k(x_k)$, then

$$\tau\left(\mathbb{E}[f(x_{k-1})]-f^*\right)\leqslant(1-\tau)\left(f^*-\mathbb{E}[D_{k-1}(x_{k-1})]\right)\leqslant(1-\tau)^k\left(f^*-D_0(x_0)\right). \tag{2.17}$$

This gives (2.7). Moreover,

$$\begin{aligned} &\mathbb{E}[f(x_k)-D_k(x_k)] \\ =&\mathbb{E}[f(x_k)-f^*]+\mathbb{E}[f^*-D_k(x_k)] \\ \leq&\frac{1}{\tau}(1-\tau)^{k+1}(f^*-D_0(x_0))+(1-\tau)^k(f^*-D_0(x_0)) \quad \text{(From 2.17 and 2.16)} \\ =&\frac{1}{\tau}(1-\tau)^k(f^*-D_0(x_0)). \end{aligned}$$

This gives (6). We conclude the proof by appropriately choosing the parameter δ which determines τ. Given the expression of the linear convergence parameter τ in (2.14), we choose δ to maximize τ, yielding

$$\delta=\min\left\{1,\frac{n\mu}{2(L-\mu)}\right\}.$$

By distinguishing the two cases, we have

1. when $\frac{n\mu}{2(L-\mu)}\leqslant 1$, we have $\delta=\frac{n\mu}{2(L-\mu)}$ and $\tau=\frac{\mu}{4(L-\mu)}$.
2. when $1\leqslant\frac{n\mu}{2(L-\mu)}$, we have $\delta=1$ and $\tau=\frac{1}{n}-\frac{L-\mu}{n^2\mu}\geqslant\frac{1}{2n}$.

Therefore, $\tau\geq\min\left(\frac{1}{2n},\frac{\mu}{4(L-\mu)}\right)$, which concludes the theorem. □

2.4 Implementation details and conclusions

In practice, the Prox-MISO algorithm provides a convenient stopping criterion, using the duality gap $f(x_k)-D_k(x_k)$. In order to keep track of the value $D_k(x_k)$ efficiently, we introduce an additional list of scalars $(C_i)_{i\in[1,n]}$ to be kept in memory. Intuitively, each C_i represents the contribution of d_i^k to the value $D_k(x_k)$ and only one of them is updated at each iteration. More precisely, we define C_i by rewriting the quadratic surrogate d_i^k as

$$d_i^k(x)=c_i^k+\frac{\mu}{2}\|x-z_i^k\|^2=C_i^k-\mu\langle x,z_i^k\rangle+\frac{\mu}{2}\|x\|^2, \tag{2.18}$$

where $C_i^k = c_i^k + \frac{\mu}{2}\|z_i^k\|^2$. When the incremental surrogate d_i is updated, the corresponding C_i^k is also updated following the rule

$$C_i^k = (1-\delta)C_i^{k-1} + \delta\left(f_i(x_{k-1}) - \langle \nabla f_i(x_{k-1}), x_{k-1}\rangle + \frac{\mu}{2}\|x_{k-1}\|^2\right).$$

This yields the practical implementation Algorithm 2.

Algorithm 2 Prox-MISO with stopping criterion.

input $(z_i^0, c_i^0)_{i=1,\dots,n}$ such that (A1) holds; ε (target accuracy);

initialize $\bar{z}_0 = \frac{1}{n}\sum_{i=1}^n z_i^0$ and $x_0 = \mathrm{prox}_{\psi/\mu}[\bar{z}_0]$, set $C_i^0 = c_i^0 + \frac{\mu}{2}\|\bar{z}_0\|^2$ for all i in $[1,n]$;

Define $\delta = \min\left(1, \frac{\mu n}{2(L-\mu)}\right)$ and $k = 0$;

while $\frac{1}{n}\sum_{i=1}^n (f_i(x_k) - C_i^k) + \mu\langle x_k, \bar{z}_k\rangle - \frac{\mu}{2}\|x_k\|^2 > \varepsilon$ **do**

  for $l = 1, \dots, n$ **do**

    $k \leftarrow k+1$;

    randomly pick up an index i_k in $[1,n]$ and perform the update (2.5);

    in the same time, update

$$C_i^k = \begin{cases} (1-\delta)C_i^{k-1} + \delta\left(f_i(x_{k-1}) - \langle \nabla f_i(x_{k-1}), x_{k-1}\rangle + \frac{\mu}{2}\|x_{k-1}\|^2\right) & \text{if} \quad i = i_k \\ C_i^{k-1} & \text{otherwise} \end{cases}. \tag{2.19}$$

  end for

end while

output x_N (final estimate such that $f(x_N) - f^* \leq \varepsilon$).

To explain the stopping criterion in Algorithm 2, we remark that the entire surrogate D_k is the sum of d_i^k and ψ, leading to the expression

$$D_k(x) = \left(\frac{1}{n}\sum_{i=1}^n C_i^k\right) - \mu\langle x, \bar{z}_k\rangle + \frac{\mu}{2}\|x\|^2 + \psi(x),$$

Thus,

$$f(x_k) - D_k(x_k) = \left(\frac{1}{n}\sum_{i=1}^n f_i(x_k) - C_i^k\right) + \mu\langle x_k, \bar{z}_k\rangle - \frac{\mu}{2}\|x_k\|^2,$$

which justifies the stopping criterion. Since computing $F(x_k)$ requires scanning all the data points, the criterion is only computed every n iterations.

Conclusions. We have proposed a proximal variant of the MISO algorithm which is able to deal with non-smooth regularization. Moreover, we remove the big data condition $n \geq \frac{2L}{\mu}$ required in the original MISO algorithm. We show that Prox-MISO is linearly convergent

when the problem is strongly convex and the resulting linear rate is similar to other variance reduction based incremental algorithms. Extensions to non-uniform sampling strategy and to stochastic setting are developed later by Bietti and Mairal [2016]. However, we remark that our algorithm is restricted to strongly convex problems and the obtained convergence rate is not optimal. The generic acceleration scheme Catalyst that we are presenting in the next chapter will solve both of these issues: a) Catalyst provides a way to apply Prox-MISO or other incremental algorithms to non strongly convex problems; b) The obtained convergence rate of applying Catalyst is optimal up to a logarithmic constant, in both strongly convex and non-strongly convex settings. The practical performance of Prox-MISO algorithm is similar to other incremental methods. Thus, we leave the experimental evaluation to the next chapter, at the same time as the evaluation of the acceleration scheme.

Chapter 3

Catalyst acceleration

Chapter abstract:
We introduce a generic scheme for accelerating gradient-based optimization methods in the sense of Nesterov. Our approach, called Catalyst, builds upon the inexact accelerated proximal point algorithm for minimizing a convex objective function, and consists of approximately solving a sequence of well-chosen auxiliary problems, leading to faster convergence. One of the key to achieve acceleration in theory and in practice is to solve these sub-problems with appropriate accuracy by using the right stopping criterion and warm start strategy. In this chapter, we discuss these practical issues and also provide a global complexity analysis. We show that our approach applies to a large class of algorithms, including gradient descent, block coordinate descent, incremental algorithms such as SAG, SAGA, SDCA, SVRG, Finito/MISO, and their proximal variants. For all of these methods, we provide acceleration and explicit support for non-strongly convex objectives. We conclude with extensive experiments showing that acceleration is useful in practice, especially for ill-conditioned problems.

The material of this chapter is based on the following paper in preparation:

H. Lin, J. Mairal, and Z. Harchaoui. Catalyst Acceleration for Gradient-Based Optimization: from Theory to Practice. 2017.

This paper extends the original conference publication:

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

The code for reproducing the figures in this chapter is publicly available at: https://github.com/hongzhoulin89/Catalyst-QNing

3.1 Introduction

A large number of machine learning and signal processing problems are formulated as the minimization of a convex composite objective function $f : \mathbb{R}^p \to \mathbb{R}$:

$$\min_{x \in \mathbb{R}^p} \Big\{ f(x) \triangleq f_0(x) + \psi(x) \Big\}, \tag{3.1}$$

where f_0 is convex and L-smooth[1], and ψ is convex but may not be differentiable. In statistics or machine learning, the variable x may represent model parameters, and the role of f_0 is to ensure that the estimated parameters fit some observed data. Specifically, f_0 is often a large sum of functions and (3.1) is a regularized empirical risk

$$\min_{x \in \mathbb{R}^p} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\}. \tag{3.2}$$

Each term $f_i(x)$ measures the fit between x and a data point indexed by i, whereas the function ψ acts as a regularizer; it is typically chosen to be the squared ℓ_2-norm, which is smooth, or to be a non-differentiable penalty such as the ℓ_1-norm or another sparsity-inducing norm [Bach et al., 2012]. Composite minimization also encompasses constrained minimization when considering extended-valued indicator functions ψ that may take the value $+\infty$ outside of a convex set and 0 inside [see Hiriart-Urruty and Lemaréchal, 1996].

In this chapter, we provide a generic framework that is able to accelerate gradient-based or first-order methods, with a particular focus on large sums of functions. By "accelerating", we mean generalizing a mechanism invented by Nesterov [2004] that improves the convergence rate of the gradient descent algorithm. More precisely, when $\psi = 0$, gradient descent steps produce iterates $(x_k)_{k \geq 0}$ such that $f(x_k) - f^* \leq \varepsilon$ in $O(1/\varepsilon)$ iterations, where f^* denotes the minimum value of f. Furthermore, when the objective f is μ-strongly convex, the previous iteration-complexity becomes $O((L/\mu)\log(1/\varepsilon))$, which is proportional to the condition number L/μ. However, these rates were shown to be suboptimal for the class of first-order methods, and a simple strategy of taking the gradient step at a well-chosen point different from x_k yields the optimal complexity—$O(1/\sqrt{\varepsilon})$ for the convex case and $O(\sqrt{L/\mu}\log(1/\varepsilon))$ for the μ-strongly convex one [Nesterov, 1983]. Later, this acceleration technique was extended to deal with non-differentiable penalties ψ for which the proximal operator defined below is easy to compute [Beck and Teboulle, 2009, Nesterov, 2013].

$$\operatorname{prox}_{\psi}(x) \triangleq \operatorname*{arg\,min}_{z \in \mathbb{R}^p} \left\{ \psi(z) + \frac{1}{2}\|x - z\|^2 \right\}, \tag{3.3}$$

where $\|.\|$ denotes the Euclidean norm.

For machine learning problems involving a large sum of n functions, a recent effort has been devoted to developing fast incremental algorithms such as SAG [Schmidt et al., 2017], SAGA

[1] We call a function L-smooth when it is differentiable and its gradient is L-Lipschitz continuous.

[Defazio et al., 2014a], SDCA [Shalev-Shwartz and Zhang, 2012], SVRG [Johnson and Zhang, 2013, Xiao and Zhang, 2014], or Finito/MISO [Defazio et al., 2014b, Mairal, 2015], which can exploit the particular structure (3.2). Unlike full gradient approaches, which require computing and averaging n gradients $(1/n)\sum_{i=1}^n \nabla f_i(x)$ at every iteration, incremental techniques have a cost per-iteration that is independent of n. The price to pay is the need to store a moderate amount of information regarding past iterates, but the benefits may be significant in terms of computational complexity. More precisely, in order to achieve an ε-accurate solution for a μ-strongly convex objective, the number of gradient evaluations required by these methods is given by $O\left(\left(n + \frac{\bar{L}}{\mu}\right)\log(\frac{1}{\varepsilon})\right)$, where $\bar{L}$ is either the maximum Lipschitz constant across the gradients ∇f_i, or the average value, depending on the algorithm variant considered. Unless there is a big mismatch between $\bar{L}$ and L (global Lipschitz constant for the sum of gradients), incremental approaches significantly outperform the full gradient method whose complexity in terms of gradient evaluations is given by $O\left(n\frac{L}{\mu}\log(\frac{1}{\varepsilon})\right)$. Yet, these incremental approaches do not use Nesterov's extrapolation steps and whether or not they could be accelerated was an important open question when these methods were introduced. It was indeed only known to be the case for SDCA [Shalev-Shwartz and Zhang, 2016] for strongly convex objectives.

In this chapter, we propose a *generic acceleration scheme* that provides a universal positive answer to the previous open question. By analogy with substances that increase chemical reaction rates, we call our approach "Catalyst". Given an optimization method $\mathcal{M}$ as input, Catalyst outputs an accelerated version of it, possibly the same algorithm if the method $\mathcal{M}$ is already optimal. The sole requirement on the method in order to achieve acceleration is that it should have linear convergence rate for strongly convex problems. This is the case for full gradient methods [Beck and Teboulle, 2009, Nesterov, 2013] and block coordinate descent methods [Nesterov, 2012b, Richtárik and Takáč, 2014], which already have well-known accelerated variants. More importantly, it also applies to the previously-mentioned incremental methods, whose complexity become $\tilde{O}\left(\left(n + \sqrt{n\bar{L}/\mu}\right)\log(\frac{1}{\varepsilon})\right)$ after Catalyst acceleration, where $\tilde{O}$ hides some logarithmic dependencies on the condition number $\bar{L}/\mu$. This improves upon the non-accelerated variants, when n is smaller than the condition number. Besides, acceleration occurs regardless of the strong convexity of the objective—that is, with $\mu = 0$— which brings us to our second achievement.

Some approaches such as Finito/MISO, SDCA, or SVRG are only defined for strongly convex objectives. A classical trick to apply them to general convex functions is to add a small regularization $\varepsilon\|x\|^2$ to the objective [Shalev-Shwartz and Zhang, 2012]. The drawback of this strategy is that it requires choosing in advance the parameter ε, which is related to the target accuracy. A consequence of our work is to automatically provide a *direct support for non-strongly convex objectives*, thus removing the need of selecting ε beforehand. Moreover, an accelerated rate is directly obtained from the resulting algorithm.

A short version of this chapter has been published at the NIPS conference in 2015 [Lin et al., 2015]; in addition to simpler convergence proofs and more extensive numerical evaluation, we extend the conference paper with two important new contributions:

Moreau-Yosida smoothing interpretation. We provide a new interpretation of Catalyst as an accelerated gradient descent method with inexact gradients, applied to the Moreau-Yosida envelope of the objective function. While the link between the proximal point algorithm and the Moreau-Yosida smoothing is well known [see Themelis et al., 2016], the fact that smoothing may be useful to accelerate existing algorithms, as in Catalyst, is less intuitive and opens new perspectives. In particular, it was a key observation to our subsequent work to Catalyst, called QuickeNing [Lin et al., 2017], which relies on limited memory Quasi-Newton principles instead of Nesterov's acceleration. Besides, the point of view of Moreau-Yosida smoothing also led to new practical strategies, which we now present.

More effective stopping criteria and warm start strategies. Catalyst is a two-loop algorithm, which requires solving sub-problems in an inner-loop with enough accuracy. In this chapter, we introduce new parameter-free stopping criteria for solving these sub-problems that are more practical than those we originally presented in [Lin et al., 2015], and new warm start strategies with significantly better empirical performance.

The paper is structured as follows: We complete this introductory section with some related work in Section 3.1.1, and give a short description of the two-loop Catalyst algorithm in Section 3.1.2. Then, Section 3.2 introduces the Moreau-Yosida regularization and its inexact variant. In Section 3.3, we introduce formally the main algorithm, and its convergence analysis is presented in Section 3.4. Section 3.6 is devoted to numerical experiments and Section 3.7 concludes the chapter.

3.1.1 Related Work

Catalyst can be interpreted as a variant of the proximal point algorithm [Rockafellar, 1976, Güler, 1991], which is a central concept in convex optimization, underlying augmented Lagrangian approaches, and composite minimization schemes [Bertsekas, 2015, Parikh and Boyd, 2014]. The proximal point algorithm consists of solving (3.1) by minimizing a sequence of auxiliary problems involving a quadratic regularization term. In general, these auxiliary problems cannot be solved with perfect accuracy, and several notations of inexactness were proposed by Güler [1992], He and Yuan [2012] and Salzo and Villa [2012]. The Catalyst approach hinges upon (i) an acceleration technique for the proximal point algorithm originally introduced in the pioneer work of Güler [1992]; (ii) a more practical inexactness criterion than those proposed in the past.[2] As a result, we are able to control the rate of convergence for approximately solving the auxiliary problems with an optimization method $\mathcal{M}$. In turn, we are also able to obtain the computational complexity of the global procedure for solving (3.1), which was not possible with

[2]Note that our inexact criterion was also studied, among others, by Salzo and Villa [2012], but their analysis led to the conjecture that this criterion was too weak to warrant acceleration. Our analysis refutes this conjecture.

previous analysis [Güler, 1992, He and Yuan, 2012, Salzo and Villa, 2012]. When instantiated in different first-order optimization settings, our analysis yields systematic acceleration.

Beyond Güler [1992], several papers have inspired our work. In particular, accelerated SDCA [Shalev-Shwartz and Zhang, 2016] is an instance of an inexact accelerated proximal point algorithm, even though this was not explicitly stated in the original paper. Catalyst can be seen as a generalization of their algorithm, originally designed for stochastic dual coordinate ascent approaches. Their proof of convergence relies on different tools than ours. Specifically, we introduce an approximate sufficient descent condition, which, when satisfied, grants acceleration to any optimization method, whereas the direct proof of Shalev-Shwartz and Zhang [2016], in the context of SDCA, does not extend to non-strongly convex objectives. Another useful methodological contribution was the convergence analysis of inexact proximal gradient methods of Schmidt et al. [2011b] and Devolder et al. [2014]. Finally, similar ideas appear in the independent work [Frostig et al., 2015]. Their results overlap in part with ours, but both papers adopt different directions. Our analysis is for instance more general and provides support for non-strongly convex objectives.

Then, beyond accelerated SDCA [Shalev-Shwartz and Zhang, 2016], a lot of other works have been recently developed towards obtaining optimal incremental methods, such as APCG [Lin et al., 2014], SDPC [Zhang and Xiao, 2015b], RPDG [Lan and Zhou, 2015], Point-SAGA [Defazio, 2016] and Katyusha [Allen-Zhu, 2016]. We remark that their techniques are algorithm-specific and cannot be directly generalized into a unified scheme. However, we should mention that the complexity obtained by applying Catalyst acceleration to incremental methods only matches the optimal bound up to a logarithmic factor, which may be the price to pay for providing a generic framework.

Finally, another recent line of work has also combined smoothing techniques with outer-loop algorithms such as Quasi-Newton methods [Themelis et al., 2016, Giselsson and Fält, 2016], even though their purpose is not to accelerate existing techniques, but rather to derive new algorithms for nonsmooth optimization.

3.1.2 Overview of Catalyst

Before introducing Catalyst precisely in Section 3.3, we provide here a first overview of the algorithm and its main ideas. Catalyst is a generic approach that wraps an algorithm $\mathcal{M}$ into an accelerated one $\mathcal{A}$, which can achieve the same accuracy as $\mathcal{M}$ with reduced computational complexity. The resulting method $\mathcal{A}$ is an inner-outer loop construct, presented in Algorithm 3, where in the *inner loop* the method $\mathcal{M}$ is called to solve an auxiliary strongly-convex optimization problem, and where in the *outer loop* the sequence of iterates produced by $\mathcal{M}$ are *extrapolated* for faster convergence. There are therefore three main ingredients in Catalyst: a) a smoothing technique that produces strongly-convex sub-problems; b) an extrapolation technique to accelerate the convergence; c) a balancing principle to optimally tune the inner and outer computations.

Algorithm 3 Catalyst - Overview

input initial estimate x_0 in $\mathbb{R}^p$, smoothing parameter κ, optimization method $\mathcal{M}$.

1: Initialize $y_0 = x_0$.

2: **while** the desired accuracy is not achieved **do**

3: Find x_k using $\mathcal{M}$

$$x_k \approx \arg\min_{x \in \mathbb{R}^p} \left\{ h_k(x) \triangleq f(x) + \frac{\kappa}{2}\|x - y_{k-1}\|^2 \right\}. \tag{3.4}$$

4: Compute y_k using an extrapolation step, with β_k in $(0, 1)$

$$y_k = x_k + \beta_k(x_k - x_{k-1}).$$

5: **end while**

output x_k (final estimate).

Smoothing by infimal convolution Catalyst can be used on any algorithm $\mathcal{M}$ that enjoys a linear-convergence guarantee when minimizing strongly-convex objectives. However the objective at hand may be poorly conditioned or even might not be strongly convex.

In Catalyst, we use $\mathcal{M}$ to *approximately minimize* an auxiliary objective h_k at iteration k, defined in (3.4), which is strongly convex and has better conditioning than f. The classical notion of Moreau envelope allows to build, using smoothing by infimal convolution with a quadratic, a well-conditioned convex function F from any poorly-conditioned convex function f (see Section 3.3 for a refresher on Moreau envelopes). We shall show in Section 3.3 that a notion of *approximate Moreau envelope* allows to define precisely the first-order information collected when approximately minimizing the auxiliary objective.

Extrapolation by Nesterov acceleration Catalyst uses an extrapolation scheme "à la Nesterov" to build a sequence $(y_k)_{k\geq 0}$ updated as

$$y_k = x_k + \beta_k(x_k - x_{k-1}) \ ,$$

where $(\beta_k)_{k\geq 0}$ is a positive decreasing sequence, which we define in Section 3.3.

We shall show in Section 3.4 that we can get faster rates of convergence thanks to this extrapolation step when the smoothing parameter κ, the inner-loop stopping criterion, and the sequence $(\beta_k)_{k\geq 0}$ are carefully built.

Balancing inner and outer complexities The optimal balance between inner loop and outer loop complexity derives from the complexity bounds established in Section 3.4. Given a target accuracy and a condition number estimate of f, our bounds dictate a choice of κ that gives the optimal setting for the inner-loop stopping criterion and all technical quantities

<table>
<tr><th></th><th colspan="2">Without Catalyst</th><th colspan="2">With Catalyst</th></tr>
<tr><th></th><th>$\mu > 0$</th><th>$\mu = 0$</th><th>$\mu > 0$</th><th>$\mu = 0$</th></tr>
<tr><td>FG</td><td>$O\left(n\frac{L_f}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$</td><td>$O\left(n\frac{L_f}{\varepsilon}\right)$</td><td>$\tilde{O}\left(n\sqrt{\frac{L_f}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$</td><td>$\tilde{O}\left(n\sqrt{\frac{L_f}{\varepsilon}}\right)$</td></tr>
<tr><td>SAG/SAGA</td><td rowspan="4">$O\left(\left(n+\frac{\bar{L}}{\mu}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$</td><td>$O\left(n\frac{\bar{L}}{\varepsilon}\right)$</td><td rowspan="4">$\tilde{O}\left(\left(n+\sqrt{\frac{n\bar{L}}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$</td><td rowspan="4">$\tilde{O}\left(\sqrt{\frac{n\bar{L}}{\varepsilon}}\right)$</td></tr>
<tr><td>MISO</td><td rowspan="3">not avail.</td></tr>
<tr><td>SDCA</td></tr>
<tr><td>SVRG</td></tr>
<tr><td>Acc-FG</td><td>$O\left(n\sqrt{\frac{L_f}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right)$</td><td>$O\left(n\frac{L_f}{\sqrt{\varepsilon}}\right)$</td><td colspan="2" rowspan="2">no acceleration</td></tr>
<tr><td>Acc-SDCA</td><td>$\tilde{O}\left(\left(n+\sqrt{\frac{n\bar{L}}{\mu}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$</td><td>not avail.</td></tr>
</table>

Table 3.1 – Comparison of rates of convergence, before and after the Catalyst acceleration, in the strongly-convex and non strongly-convex cases, respectively. The notation $\tilde{O}$ hides logarithmic factors. The constant L_f is the global Lipschitz constant of the gradient with respect to the entire objective function, while $\bar{L}$ is the average Lipschitz constants of the gradients ∇f_i, or the maximum value, depending on the algorithm's variants considered.

involved in the algorithm. We shall demonstrate in particular the power of an appropriate warm start strategy to achieve near-optimal complexity.

Overview of the complexity results Finally, we provide in Table 3.1 a brief overview of the complexity results obtained from the Catalyst acceleration, when applied to various optimization methods $\mathcal{M}$ for minimizing a large finite sum of n functions. Note that the complexity results obtained with Catalyst are optimal, up to some logarithmic factors [see Agarwal and Bottou, 2015, Arjevani and Shamir, 2016, Woodworth and Srebro, 2016].

3.2 The Moreau-Yosida envelope and its inexact variant

We now briefly recall the definition of the Moreau-Yosida envelope [Moreau, 1962, Yosida, 1980], which we have already introduced in Section 1.6 and which plays a key role for understanding the Catalyst acceleration.

More precisely, the Moreau-Yosida envelope results from the infimal convolution of f with a quadratic penalty:

$$F(x) \triangleq \min_{z\in\mathbb{R}^p}\left\{f(z)+\frac{\kappa}{2}\|z-x\|^2\right\}, \tag{3.5}$$

where κ is the positive regularization parameter. Note that the proximal operator is then defined as the unique minimizer of the problem—that is,

$$p(x) \triangleq \operatorname{prox}_{f/\kappa}(x) = \arg\min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|z - x\|^2 \right\}.$$

Unlike the proximal operator of ψ defined in (3.3), $p(x)$ does not admit a closed form in general. Therefore, computing it requires to solve a sub-problem to high accuracy with some iterative algorithm.

In Section 1.6, we discussed basic properties of the Moreau-Yosida envelope. In particular, minimizing F and f is equivalent in the sense that both optimal function values and solution sets coincide. F is always $L_F = \kappa$-smooth and its gradient is given by

$$\nabla F(x) = \kappa(x - p(x)). \tag{3.6}$$

Finally, if f is μ-strongly convex, then F is μ_F-strongly convex with $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$. Note that F can then be arbitrarily well conditioned when f is μ-strongly convex, by choosing a small value of κ.

3.2.1 A Fresh Look at Catalyst

We have also seen that first-order methods applied to F provide us several known algorithms; gradient descent on F yields the proximal point algorithm [Martinet, 1970, Rockafellar, 1976] and Nesterov's accelerated method yields the the accelerated proximal point algorithm of Güler [1992]. While these algorithms are conceptually elegant, each update requires to evaluate the gradient of F, and thus to compute proximal operator $p(x)$ exactly. Unless a closed form is available, which is almost never the case, an iterative algorithm is required, which yields the two-loop scheme (see Algorithm 3). Catalyst can then be interpreted as an accelerated gradient descent method that computes inexact sub-problems using an optimization method $\mathcal{M}$:

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

where β_{k+1} is Nesterov's extrapolation parameter [Nesterov, 2004]. When the minimization is exact, $x_{k+1} = p(y_k) = y_k - \frac{1}{L_F}\nabla F(y_k)$, and we recover a gradient step computed at the extrapolated point y_k. The main challenge that will be addressed in Section 3.3 is how to control the complexity of the inner-loop minimization.

3.2.2 The Inexact Moreau-Yosida Envelope

Since Catalyst uses the Moreau-Yoside envelope with inexact gradient, it is natural to introduce inexactness criteria. We start with the most natural one.

Inexactness through absolute accuracy. The following definition introduces a set of approximate proximal operators, which we will use in Catalyst. Given a proximal center x, a smoothing parameter κ, and an accuracy $\varepsilon > 0$, we denote the set of ε-approximations of the proximal operator $p(x)$ by

$$p^{\varepsilon}(x) \triangleq \{z \in \mathbb{R}^p \quad \text{s.t.} \quad h(z) - h^* \leq \varepsilon\} \quad \text{where} \quad h(z) = f(z) + \frac{\kappa}{2}\|x - z\|^2, \tag{C1}$$

and h^* is the minimum function value of h. Checking the condition z in $p^{\varepsilon}(x)$ may typically be done by computing a duality gap, or by using another certificate of the type $h(z) - h^* \leq h(z) - d(z)$, where the lower-bound $d(z) \leq h^*$ is obtained from duality such as in SDCA [Shalev-Shwartz and Zhang, 2016]. In other methods such as MISO [Mairal, 2015], $d(z)$ may be obtained from a quadratic approximation of the objective function and may have a simple closed form. Note that a criterion based on the subdifferential may also be used as in the extension of Catalyst recently developed for non-convex optimization by Paquette et al. [2017]. Here, we focus on the convex case, where other simple criterions may be appropriate, as shown in the next lemma.

Lemma 6 (Checking the absolute accuracy criterion). *Consider a proximal center x, a smoothing parameter κ and an accuracy $\varepsilon > 0$. Consider an objective with composite form (3.1), and the function h defined in (C1). Then, define*

$$[z]_\eta = \operatorname{prox}_{\eta\psi}\left(z - \eta \nabla h_0\left(z\right)\right), \tag{3.7}$$

where h_0 is the smooth part of h, namely $\nabla h_0\left(z\right) = \nabla f_0(z) + \kappa(z - x)$ and $\eta = \frac{1}{\kappa + L}$. Then,

$$\|z - [z]_\eta\| \leq \eta\sqrt{2\kappa\varepsilon} \quad \textit{implies that} \quad [z]_\eta \in p^{\varepsilon}(x).$$

The proof is provided in Appendix B.2[see also the link with criterions based on subdifferential in Paquette et al., 2017]. Then, with an approximate proximal operator in hand, we may now obtain an approximate gradient of the Moreau-Yosida envelope, defined as

$$g(z) \triangleq \kappa(x - z), \tag{3.8}$$

for any z in $p^{\varepsilon}(x)$, according to the exact gradient formula $\nabla F(x) = \kappa(x - p(x))$. As a consequence, we may immediately draw a link

$$z \in p^{\varepsilon}(x) \quad \Longrightarrow \quad \|z - p(x)\| \leq \sqrt{\frac{2\varepsilon}{\kappa}} \quad \Longleftrightarrow \quad \|g(z) - \nabla F(x)\| \leq \sqrt{2\kappa\varepsilon}, \tag{3.9}$$

where the upper bound on $\|z - p(x)\|$ is a consequence of the strong convexity of h and the fact that $p(x)$ is its exact minimizer.

Relative error criterion. We have seen that z in $p^\varepsilon(x)$ yields an upper-bound of the form $\|g(z)-\nabla F(x)\| \leq \varepsilon'$. Another criterion that arise naturally in the convergence analysis provides instead an upper bound of the form $\|g(z) - \nabla F(x)\| \leq \delta'\|\nabla F(x)\|$ for some $\delta' > 0$—that is, providing a relative error in terms of gradient approximation.

Given a proximal center x, a smoothing parameter κ and a relative accuracy δ in $[0, 1)$, we denote the set of δ-relative approximations by

$$g^\delta(x) \triangleq \left\{ z \in \mathbb{R}^p \quad \text{s.t.} \quad h(z) - h^* \leq \frac{\delta\kappa}{2}\|x - z\|^2 \right\}, \tag{C2}$$

Criterion (C2) may be interpreted as (C1) with an adaptive ε that depends on the point z. Therefore, given a fixed z, checking if z is in $g^\delta(x)$ is equivalent to checking if z is in $p^\varepsilon(x)$ with $\varepsilon = \frac{\delta\kappa}{2}\|x - z\|^2$. Then, by following similar steps as in (3.9), notice that z in $g^\delta(x)$ implies

$$\|z - p(x)\| \leq \sqrt{\delta}\|x - z\| \leq \sqrt{\delta}(\|x - p(x)\| + \|p(x) - z\|),$$

and by rearranging, and using closed forms of $g(z)$ and $\nabla F(x)$,

$$z \in g^\delta(x) \quad \Longrightarrow \quad \|g(z) - \nabla F(x)\| \leq \delta'\|\nabla F(x)\| \quad \text{with} \quad \delta' = \frac{\sqrt{\delta}}{1 - \sqrt{\delta}},$$

which provides a relative gradient approximation.

Finally, note that Lemma 6 may be used as well to obtain a point $[z]_\eta$ in $g^\delta(x)$ after taking an extra gradient and prox steps and setting the value $\varepsilon = \frac{\delta\kappa}{2}\|x - z\|$.

A few remarks on related works. Inexactness criteria with respect to subgradient norms have been investigated in the past, starting from the pioneer work of Rockafellar [1976] in the context of the inexact proximal point algorithm. Later, different works have been dedicated to more practical inexactness criteria[3] [Auslender, 1987, Correa and Lemaréchal, 1993, Solodov and Svaiter, 2001, Fuentes et al., 2012]. Here, we provide a more intuitive point of view using the Moreau-Yosida envelope.

While the proximal point algorithm has caught a lot of attention, very few works have focused on its accelerated variant. The first accelerated proximal point algorithm with inexact gradients was proposed by Güler [1992]. Then, Salzo and Villa [2012] proposed a more rigorous convergence analysis, and more inexactness criteria, which are typically stronger than ours. In the same way, a more general inexact oracle framework has been proposed later by Devolder et al. [2014]. To achieve the Catalyst acceleration, our main effort was to propose and analyze criteria that allow us to control the complexity for finding approximate solutions of the sub-problems.

[3]Such as, duality gap, ε-subdifferential or decrease in terms of function value.

3.3 Catalyst acceleration

In this section, we introduce the Catalyst algorithm, and discuss its main features.

3.3.1 Main Algorithm

Catalyst is presented in Algorithm 4. As discussed in Section 3.2, this scheme can be interpreted as an inexact accelerated proximal point algorithm, or equivalently as an accelerated gradient descent method applied to the Moreau-Yosida envelope of the objective with inexact gradients. Since an overview has already been presented in Section 3.1.2, we now present important details to obtain acceleration in theory and in practice.

Algorithm 4 Catalyst

input Initial estimate x_0 in $\mathbb{R}^p$, smoothing parameter κ, strong convexity parameter μ, optimization method $\mathcal{M}$ and a stopping criterion based on a sequence of accuracies $(\varepsilon_k)_{k\geq 0}$, or $(\delta_k)_{k\geq 0}$, or a fixed budget T.

1: Initialize $y_0 = x_0$, $q = \frac{\mu}{\mu+\kappa}$. If $\mu > 0$, set $\alpha_0 = \sqrt{q}$, otherwise $\alpha_0 = 1$.

2: **while** the desired accuracy is not achieved **do**

3: Compute an approximate solution of the following problem with $\mathcal{M}$

$$x_k \approx \arg\min_{x\in\mathbb{R}^p} \left\{ h_k(x) \triangleq f(x) + \frac{\kappa}{2}\|x - y_{k-1}\|^2 \right\},$$

using the warm start strategy of Section 3.3.1 and one of the following stopping criteria:

(a) *a*bsolute accuracy: find x_k in $p^{\varepsilon_k}(y_{k-1})$ by using criterion (C1);

(b) *r*elative accuracy: find x_k in $g^{\delta_k}(y_{k-1})$ by using criterion (C2);

(c) *f*ixed budget: run $\mathcal{M}$ for T iterations and output x_k.

4: Update α_k in $(0, 1)$ by solving the equation

$$\alpha_k^2 = (1 - \alpha_k)\alpha_{k-1}^2 + q\alpha_k \tag{3.10}$$

5: Compute y_k with Nesterov's extrapolation step

$$y_k = x_k + \beta_k(x_k - x_{k-1}) \quad \text{with} \quad \beta_k = \frac{\alpha_{k-1}(1 - \alpha_{k-1})}{\alpha_{k-1}^2 + \alpha_k}. \tag{3.11}$$

6: **end while**

output x_k (final estimate).

Requirement: linear convergence of the method $\mathcal{M}$. One of the main characteristic of Catalyst is to apply the method $\mathcal{M}$ to strongly-convex sub-problems, without requiring strong convexity of the objective f. As a consequence, Catalyst provides direct support for convex but non-strongly objectives to $\mathcal{M}$, which may be useful to extend the scope of application of techniques that need strong convexity to operate. Yet, Catalyst requires to solve these sub-problems efficiently enough in order to control the complexity of the inner-loop computations. More precisely, when applying $\mathcal{M}$ to minimize a strongly-convex function h, we assume that $\mathcal{M}$ is able to produce a sequence of iterates $(z_t)_{t\geq 0}$ such that

$$h(z_t) - h^* \leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^t(h(z_0)-h^*), \tag{3.12}$$

where z_0 is the initial point given to $\mathcal{M}$, and $\tau_{\mathcal{M}}$ in $(0,1)$, $C_{\mathcal{M}} > 0$ are two constants. In such a case, we say that $\mathcal{M}$ admits a linear convergence rate. The quantity $\tau_{\mathcal{M}}$ controls the speed of convergence for solving the sub-problem: the larger is $\tau_{\mathcal{M}}$, the faster is convergence. For a given algorithm $\mathcal{M}$, the quantity $\tau_{\mathcal{M}}$ depends usually on the condition number of h. For instance, for the proximal gradient method and many first-order algorithms, we simply have $\tau_{\mathcal{M}} = O((\mu+\kappa)/(L+\kappa))$, as h is $(\mu+\kappa)$-strongly convex and $(L+\kappa)$-smooth. Catalyst can also be applied to randomized methods $\mathcal{M}$ that satisfy (3.12) in expectation:

$$\mathbb{E}[h(z_t) - h^*] \leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^t(h(z_0)-h^*), \tag{3.13}$$

Then, the complexity results of Section 3.4 also hold in expectation. This allows us to apply Catalyst to randomized block coordinate descent algorithms [see Richtárik and Takáč, 2014, and references therein], and some incremental algorithms such as SAG, SAGA, or SVRG. For other methods that admit a linear convergence rates in terms of duality gap, such as SDCA, MISO/Finito, Catalyst can also be applied as explained in Section 3.5.

Stopping criteria. Catalyst may be used with three types of stopping criteria for solving the inner-loop problems. We now detail them below.

(a) *absolute accuracy*: we use a sequence $(\varepsilon_k)_{k\geq 0}$ of accuracies, and stop the method $\mathcal{M}$ by using criterion (C1). Typically, (C1) requires to compute a duality gap to guarantee that $h_k(z_t) - h_k^* \leq \varepsilon_k$. Our analysis suggests the following choice for $(\varepsilon_k)_{k\geq 0}$:

- if f is μ-strongly convex, use $\alpha_0 = \sqrt{q}$ and

$$\varepsilon_k = \frac{2}{9}(1-\rho)^k(f(x_0)-f^*) \quad \text{with} \quad \rho < \sqrt{q}.$$

- if f is convex but not strongly convex, use $\alpha_0 = 1$.

$$\varepsilon_k = \frac{2(f(x_0)-f^*)}{9(k+2)^{4+\gamma}} \quad \text{with} \quad \gamma > 0.$$

Typically, $\gamma = 0.1$ and $\rho = 0.9\sqrt{q}$ are reasonable choices, both in theory and in practice. Of course, the quantity $f(x_0) - f^*$ is unknown, but it may be replaced safely by an upper bound such as $f(x_0)$ if f is non-negative or by a duality gap.

(b) *relative accuracy*: To use criterion (C2), our analysis suggests the following choice for the sequence $(\delta_k)_{k\geq 0}$:

- if f is μ-strongly convex,
$$\delta_k = \frac{\sqrt{q}}{2 - \sqrt{q}}.$$
- if f is convex but not strongly convex,
$$\delta_k = \frac{1}{(k+1)^2},$$
and in both cases, the value of α_0 is the same as for criterion (C1).

With no a priori knowledge required about the objective function f, these relative accuracy criteria (b) are sometimes simpler to use than criteria (a).

(c) *fixed budget*: Finally, the simplest way of using Catalyst is to fix in advance the number T of iterations of the method $\mathcal{M}$ for solving the sub-problems without checking any optimality criterion. Whereas our analysis provides theoretical budgets that are compatible with this strategy, we found them to be pessimistic and impractical. Instead, we propose an aggressive strategy for incremental methods that simply consists of setting $T = n$. This setting was called the "one-pass" strategy in the original Catalyst paper [Lin et al., 2015].

Warm-starts in inner loops. Besides linear convergence rate, an adequate warm start strategy needs to be used to guarantee that the sub-problems will be solved in reasonable computational time. Specifically, the following choices arise from the convergence analysis that will be detailed in Section 3.4. Consider a sub-problem consisting of minimizing the function

$$h_k(z) \triangleq f(z) + \frac{\kappa}{2}\|z - y_{k-1}\|^2.$$

Then, $\mathcal{M}$ will produce a sequence of iterates $(z_t)_{t\geq 0}$ with initial point z_0, which enjoys linear convergence (3.12) or (3.13). To achieve the Catalyst acceleration, the choice of z_0 is critical. Here, we propose the following rules, which are both compatible with the theory, and more effective in practice than the one introduced in the original Catalyst paper [Lin et al., 2015].

(a) when using criterion (C1) to find x_k in $p^{\varepsilon_k}(y_{k-1})$, choose

- if f is smooth ($\psi = 0$), then choose $z_0 = x_{k-1} + \frac{\kappa}{\kappa+\mu}(y_{k-1} - y_{k-2})$.
- if f is composite as in (3.1), then define $w_0 = x_{k-1} + \frac{\kappa}{\kappa+\mu}(y_{k-1} - y_{k-2})$ and

$$z_0 = \text{prox}_{\eta\psi}(w_0 - \eta g) \quad \text{with} \quad g = \nabla f_0(w_0) + \kappa(w_0 - y_{k-1}) \quad \text{and} \quad \eta = \frac{1}{L+\kappa}. \tag{3.14}$$

At the cost of an extra function evaluation, we also noticed that the warm start strategy can be improved by choosing the starting point among x_{k-1} and (3.14) that yields the smallest function value for h.

(b) when using criteria (C2) to find x_k in $g^{\delta_k}(y_{k-1})$,

- if f is smooth ($\psi = 0$), then choose $z_0 = y_{k-1}$.
- if f is composite as in (3.1), then choose

$$z_0 = \text{prox}_{\eta\psi}(y_{k-1} - \eta\nabla f_0(y_{k-1})) \quad \text{with} \quad \eta = \frac{1}{L+\kappa}.$$

(c) when using a fixed budget T, choose the same warm start strategy as in (a).

Note that the original paper about Catalyst [Lin et al., 2015] only considers criterion (C1) with the warm start rule $z_0 = x_{k-1}$, which does not perform as well as the ones proposed here.

Optimal balance: choice of parameter κ. Finally, the last ingredient is to find an optimal balance between the inner-loop (for solving each sub-problem) and outer-loop computations. To do so, we minimize our global complexity bounds with respect to the value of κ. As studied in the experimental section, this strategy turns out to be reasonable in practice. Then, as shown in the theoretical section, the resulting rule of thumb is

We select κ by maximizing the ratio $\tau_{\mathcal{M}}/\sqrt{\mu+\kappa}$.

We recall that $\tau_{\mathcal{M}}$ characterizes how fast $\mathcal{M}$ solves the sub-problems, according to (3.12); typically, $\tau_{\mathcal{M}}$ depends on the condition number $\frac{L+\kappa}{\mu+\kappa}$ and is a function of κ.[4] In Table 3.2, we illustrate the choice of κ for different methods. Note that the resulting rule for incremental methods is very simple for the practitioner: select κ such that the condition number $\frac{\bar{L}+\kappa}{\mu+\kappa}$ is of the order of n; then, the inner-complexity becomes $O(n\log(1/\varepsilon))$.

[4] Note that the rule for the non strongly convex case, denoted here by $\mu = 0$, slightly differs from Lin et al. [2015] and results from a tighter complexity analysis.

Method $\mathcal{M}$	Inner-complexity	$\tau_{\mathcal{M}}$	Choice for κ
FG	$O\left(n\frac{L+\kappa}{\mu+\kappa}\log\left(\frac{1}{\varepsilon}\right)\right)$	$\propto \frac{\mu+\kappa}{L+\kappa}$	$L-2\mu$
SAG/SAGA/SVRG	$O\left(\left(n+\frac{\bar{L}+\kappa}{\mu+\kappa}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$	$\propto \frac{\mu+\kappa}{n(\mu+\kappa)+\bar{L}+\kappa}$	$\frac{\bar{L}-\mu}{n+1}-\mu$

Table 3.2 – Example of choices of the parameter κ for the full gradient (FG) and incremental methods SAG/SAGA/SVRG. See Table 3.1 for details about the complexity of the methods.

3.4 Convergence and complexity analysis

We now present the complexity analysis of Catalyst. In Section 3.4.1, we analyze the convergence rate of the outer loop, regardless of the complexity for solving the sub-problems. Then, we analyze the complexity of the inner-loop computations for our various stopping criteria and warm start strategies in Section 3.4.2. Section 3.4.3 combines the outer- and inner-loop analysis to provide the global complexity of Catalyst applied to a given optimization method $\mathcal{M}$.

3.4.1 Complexity Analysis for the Outer-Loop

The complexity analysis of the first variant of Catalyst we presented in [Lin et al., 2015] used a tool called "estimate sequence", which was introduced by Nesterov [2004]. Here, we provide a simpler proof. We start with criterion (C1), before extending the result to (C2).

Analysis for Criterion (C1)

The next theorem describes how the errors $(\varepsilon_k)_{k\geq 0}$ accumulate in Catalyst.

Theorem 7 (Convergence of outer-loop for criterion (C1)). *Consider the sequences $(x_k)_{k\geq 0}$ and $(y_k)_{k\geq 0}$ produced by Algorithm 4, assuming that x_k is in $p^{\varepsilon_k}(y_{k-1})$ for all $k \geq 1$, Then,*

$$f(x_k) - f^* \leq A_{k-1}\left(\sqrt{(1-\alpha_0)(f(x_0)-f^*) + \frac{\gamma_0}{2}\|x^*-x_0\|^2} + 3\sum_{j=1}^{k}\sqrt{\frac{\varepsilon_j}{A_{j-1}}}\right)^2,$$

where

$$\gamma_0 = (\kappa+\mu)\alpha_0(\alpha_0-q) \quad \textit{and} \quad A_k = \prod_{j=1}^{k}(1-\alpha_j). \tag{3.15}$$

Before we prove this theorem, we note that by setting $\varepsilon_k = 0$ for all k, the speed of convergence of $f(x_k) - f^*$ is driven by the sequence $(A_k)_{k\geq 0}$. Lemma 2.2.4 of Nesterov [2004], which we recall below, provides us this speed.

Lemma 7 (**Lemma 2.2.4 of Nesterov 2004**). *Consider the quantities γ_0, A_k defined in (3.15) and the α_k's defined in Algorithm 4. Then, if $\gamma_0 \geq \mu$,*

$$A_k \leq \min\left\{(1-\sqrt{q})^k, \frac{4}{\left(2+k\sqrt{\frac{\gamma_0}{\kappa}}\right)^2}\right\}.$$

For non-strongly convex objectives, A_k follows the classical accelerated $O(1/k^2)$ rate of convergence, whereas it achieves a linear convergence rate for the strongly convex case. Note that the quantity $q = \frac{\mu}{\mu+\kappa}$ is simply the inverse square root of the condition number of the Moreau-Yosida regularization F of the objective f. We now provide the proof of the theorem below.

Proof. We start by defining an approximate sufficient descent condition inspired by a remark of Chambolle and Pock [2015] regarding accelerated gradient descent methods. A related condition was also used by Paquette et al. [2017] in the context of non-convex optimization. □

Approximate sufficient descent condition. Let us define the function $h_k(x) = f(x) + \frac{\kappa}{2}\|x - y_k\|^2$. Since $p(y_k)$ is the unique minimizer of h_k, we may start with a strong convexity inequality: for any $k \geq 1$, for all x in $\mathbb{R}^d$ and $\theta_k \geq 0$,

$$\begin{aligned}
h_k(x) &\geq h_k^* + \frac{\kappa+\mu}{2}\|x - p(y_{k-1})\|^2 \\
&\geq h_k^* + \frac{\kappa+\mu}{2}(1-\theta_k)\|x - x_k\|^2 - \frac{\kappa+\mu}{2}\left(\frac{1}{\theta_k} - 1\right)\|x_k - p(y_{k-1})\|^2 \\
&\geq h_k(x_k) - \varepsilon_k + \frac{\kappa+\mu}{2}(1-\theta_k)\|x - x_k\|^2 - \frac{\kappa+\mu}{2}\left(\frac{1}{\theta_k} - 1\right)\|x_k - p(y_{k-1})\|^2,
\end{aligned}$$

where the $(\mu+\kappa)$-strong convexity of h_k is used in the first inequality; Lemma 16 is used in the second inequality, and the last one uses the relation $h_k(x_k) - h_k^* \leq \varepsilon_k$. Moreover, when $\theta_k \geq 1$, the last term is non-negative and

$$h_k(x) \geq h_k(x_k) - \varepsilon_k + \frac{\kappa+\mu}{2}(1-\theta_k)\|x - x_k\|^2.$$

If instead $\theta_k \leq 1$, the coefficient $\frac{1}{\theta_k} - 1$ is non-negative and we have

$$-\frac{\kappa+\mu}{2}\left(\frac{1}{\theta_k} - 1\right)\|x_k - p(y_{k-1})\|^2 \geq -\left(\frac{1}{\theta_k} - 1\right)(h_k(x_k) - h_k^*) \geq -\left(\frac{1}{\theta_k} - 1\right)\varepsilon_k.$$

In this case, we have

$$h_k(x) \geq h_k(x_k) - \frac{\varepsilon_k}{\theta_k} + \frac{\kappa+\mu}{2}(1-\theta_k)\|x - x_k\|^2.$$

As a result, we have for all value of $\theta_k \geq 0$,

$$h_k(x) \geq h_k(x_k) + \frac{\kappa+\mu}{2}(1-\theta_k)\|x-x_k\|^2 - \frac{\varepsilon_k}{\min\{1,\theta_k\}}.$$

After using the definition of h_k, we then obtain the approximate descent condition

$$f(x_k) + \frac{\kappa}{2}\|x_k - y_{k-1}\|^2 + \frac{\kappa+\mu}{2}(1-\theta_k)\|x-x_k\|^2 \leq f(x) + \frac{\kappa}{2}\|x-y_{k-1}\|^2 + \frac{\varepsilon_k}{\min\{1,\theta_k\}}. \quad (3.16)$$

Definition of the Lyapounov function. we introduce a sequence $(S_k)_{k\geq 0}$ that will act as a Lyapounov function, with

$$S_k = (1-\alpha_k)(f(x_k) - f^*) + \alpha_k \frac{\kappa\eta_k}{2}\|x^* - v_k\|^2. \quad (3.17)$$

where x^* is a minimizer of f, $(v_k)_{k\geq 0}$ is a sequence defined by $v_0 = x_0$ and

$$v_k = x_k + \frac{1-\alpha_{k-1}}{\alpha_{k-1}}(x_k - x_{k-1}) \quad \text{for } k \geq 1,$$

and $(\eta_k)_{k\geq 0}$ is an auxiliary quantity defined by

$$\eta_k = \frac{\alpha_k - q}{1-q}.$$

The way we introduce these variables allow us to write the following relationship,

$$y_k = \eta_k v_k + (1-\eta_k)x_k, \quad \text{for all } k \geq 0,$$

which is obtained after simple calculation. Then, we have the following relations for all $k \geq 1$ and $z_k = \alpha_{k-1}x^* + (1-\alpha_{k-1})x_{k-1}$.

$$\begin{aligned} f(z_k) &\leq \alpha_{k-1}f^* + (1-\alpha_{k-1})f(x_{k-1}) - \frac{\mu\alpha_{k-1}(1-\alpha_{k-1})}{2}\|x^* - x_{k-1}\|^2; \\ z_k - x_k &= \alpha_{k-1}(x^* - v_k), \end{aligned}$$

and also the following one

$$\begin{aligned} \|z_k - y_{k-1}\|^2 &= \|(\alpha_{k-1} - \eta_{k-1})(x^* - x_{k-1}) + \eta_{k-1}(x^* - v_{k-1})\|^2 \\ &= \alpha_{k-1}^2 \left\| \left(1 - \frac{\eta_{k-1}}{\alpha_{k-1}}\right)(x^* - x_{k-1}) + \frac{\eta_{k-1}}{\alpha_{k-1}}(x^* - v_{k-1}) \right\|^2 \\ &\leq \alpha_{k-1}^2 \left(1 - \frac{\eta_{k-1}}{\alpha_{k-1}}\right)\|x^* - x_{k-1}\|^2 + \alpha_{k-1}^2 \frac{\eta_{k-1}}{\alpha_{k-1}}\|x^* - v_{k-1}\|^2 \\ &= \alpha_{k-1}(\alpha_{k-1} - \eta_{k-1})\|x^* - x_{k-1}\|^2 + \alpha_{k-1}\eta_{k-1}\|x^* - v_{k-1}\|^2, \end{aligned}$$

where we used the convexity of the norm and the fact that $\eta_k \leq \alpha_k$. Using the previous relations in (3.16) with $x = z_k = \alpha_{k-1}x^* + (1-\alpha_{k-1})x_{k-1}$, gives for all $k \geq 1$,

$$\begin{aligned} f(x_k) + \frac{\kappa}{2}\|x_k - y_{k-1}\|^2 + \frac{\kappa+\mu}{2}(1-\theta_k)\alpha_{k-1}^2\|x^* - v_k\|^2 \\ \leq \alpha_{k-1}f^* + (1-\alpha_{k-1})f(x_{k-1}) - \frac{\mu}{2}\alpha_{k-1}(1-\alpha_{k-1})\|x^* - x_{k-1}\|^2 \\ + \frac{\kappa\alpha_{k-1}(\alpha_{k-1}-\eta_{k-1})}{2}\|x^* - x_{k-1}\|^2 + \frac{\kappa\alpha_{k-1}\eta_{k-1}}{2}\|x^* - v_{k-1}\|^2 + \frac{\varepsilon_k}{\min\{1,\theta_k\}}. \end{aligned}$$

Remark that for all $k \geq 1$,

$$\alpha_{k-1} - \eta_{k-1} = \alpha_{k-1} - \frac{\alpha_{k-1}-q}{1-q} = \frac{q(1-\alpha_{k-1})}{1-q} = \frac{\mu}{\kappa}(1-\alpha_{k-1}),$$

and the quadratic terms involving $x^* - x_{k-1}$ cancel each other. Then, after noticing that for all $k \geq 1$,

$$\eta_k\alpha_k = \frac{\alpha_k^2 - q\alpha_k}{1-q} = \frac{(\kappa+\mu)(1-\alpha_k)\alpha_{k-1}^2}{\kappa},$$

which allows us to write

$$f(x_k) - f^* + \frac{\kappa+\mu}{2}\alpha_{k-1}^2\|x^* - v_k\|^2 = \frac{S_k}{1-\alpha_k}. \tag{3.18}$$

We are left, for all $k \geq 1$, with

$$\frac{1}{1-\alpha_k}S_k \leq S_{k-1} + \frac{\varepsilon_k}{\min\{1,\theta_k\}} - \frac{\kappa}{2}\|x_k - y_{k-1}\|^2 + \frac{(\kappa+\mu)\alpha_{k-1}^2\theta_k}{2}\|x^* - v_k\|^2. \tag{3.19}$$

Control of the approximation errors for criterion (C1). Using the fact that

$$\frac{1}{\min\{1,\theta_k\}} \leq 1 + \frac{1}{\theta_k},$$

we immediately derive from equation (3.19) that

$$\frac{1}{1-\alpha_k}S_k \leq S_{k-1} + \varepsilon_k + \frac{\varepsilon_k}{\theta_k} - \frac{\kappa}{2}\|x_k - y_{k-1}\|^2 + \frac{(\kappa+\mu)\alpha_{k-1}^2\theta_k}{2}\|x^* - v_k\|^2. \tag{3.20}$$

By minimizing the right-hand side of (3.20) with respect to θ_k, we obtain the following inequality

$$\frac{1}{1-\alpha_k}S_k \leq S_{k-1} + \varepsilon_k + \sqrt{2\varepsilon_k(\mu+\kappa)}\alpha_{k-1}\|x^* - v_k\|,$$

and after unrolling the recursion,

$$\frac{S_k}{A_k} \leq S_0 + \sum_{j=1}^{k} \frac{\varepsilon_j}{A_{j-1}} + \sum_{j=1}^{k} \frac{\sqrt{2\varepsilon_j(\mu+\kappa)}\alpha_j \|x^* - v_j\|}{A_{j-1}}.$$

From Equation (3.18), the lefthand side is larger than $\frac{(\mu+\kappa)\alpha_{k-1}^2\|x^*-v_k\|^2}{2A_{k-1}}$. We may now define $u_j = \frac{\sqrt{(\mu+\kappa)}\alpha_{j-1}\|x^*-v_j\|}{\sqrt{2A_j}}$ and $a_j = 2\frac{\sqrt{\varepsilon_j}}{\sqrt{A_j}}$, and we have

$$u_k^2 \leq S_0 + \sum_{j=1}^{k} \frac{\varepsilon_j}{A_{j-1}} + \sum_{j=1}^{k} a_j u_j \quad \text{for all } k \geq 1.$$

This allows us to use Lemma 17, which yields

$$\begin{aligned}\frac{S_k}{A_k} &\leq \left(\sqrt{S_0 + \sum_{j=1}^{k} \frac{\varepsilon_j}{A_{j-1}}} + 2\sum_{j=1}^{k}\sqrt{\frac{\varepsilon_j}{A_{j-1}}}\right)^2, \\ &\leq \left(\sqrt{S_0} + 3\sum_{j=1}^{k}\sqrt{\frac{\varepsilon_j}{A_{j-1}}}\right)^2\end{aligned}$$

which provides us the desired result given that $f(x_k) - f^* \leq \frac{S_k}{1-\alpha_k}$ and that $v_0 = x_0$.

We are now in shape to state the convergence rate of the Catalyst algorithm with criterion (C1), without taking into account yet the cost of solving the sub-problems. The next two propositions specialize Theorem 7 to the strongly convex case and convex, but not strongly convex cases, respectively. Their proofs are provided in Appendix B.2.

Proposition 2 (μ**-strongly convex case, criterion (C1)).**
In Algorithm 4, choose $\alpha_0 = \sqrt{q}$ *and*

$$\varepsilon_k = \frac{2}{9}(f(x_0) - f^*)(1-\rho)^k \quad \text{with} \quad \rho < \sqrt{q}.$$

Then, the sequence of iterates $(x_k)_{k\geq 0}$ *satisfies*

$$f(x_k) - f^* \leqslant \frac{8}{(\sqrt{q}-\rho)^2}(1-\rho)^{k+1}(f(x_0) - f^*).$$

Proposition 3 (**Convex case, criterion (C1)).**
When $\mu = 0$, *choose* $\alpha_0 = 1$ *and*

$$\varepsilon_k = \frac{2(f(x_0) - f^*)}{9(k+1)^{4+\gamma}} \quad \text{with} \quad \gamma > 0.$$

Then, Algorithm 4 generates iterates $(x_k)_{k\geq 0}$ *such that*

$$f(x_k) - f^* \leqslant \frac{8}{(k+1)^2}\left(\frac{\kappa}{2}\|x_0 - x^*\|^2 + \frac{4}{\gamma^2}(f(x_0) - f^*)\right).$$

Analysis for Criterion (C2)

Then, we may now analyze the convergence of Catalyst under criterion (C2), which offers similar guarantees as (C1), as far as the outer loop is concerned.

Theorem 8 (Convergence of outer-loop for criterion (C2)). *Consider the sequences $(x_k)_{k\geq 0}$ and $(y_k)_{k\geq 0}$ produced by Algorithm 4, assuming that x_k is in $g^{\delta_k}(y_{k-1})$ for all $k \geq 1$ and δ_k in $(0,1)$. Then,*

$$f(x_k) - f^* \leq \frac{A_{k-1}}{\prod_{j=1}^k (1-\delta_j)} \left((1-\alpha_0)(f(x_0) - f^*) + \frac{\gamma_0}{2}\|x_0 - x^*\|^2 \right),$$

where γ_0 and $(A_k)_{k\geq 0}$ are defined in (3.15) in Theorem 7.

Proof. x_k in $g^{\delta_k}(y_{k-1})$ is equivalent to x_k in $p^{\varepsilon_k}(y_{k-1})$ with an adaptive $\varepsilon_k = \frac{\delta_k \kappa}{2}\|x_k - y_{k-1}\|^2$. All steps of the proof of Theorem 7 hold for such values of ε_k and from (3.19), we may deduce

$$\frac{S_k}{1-\alpha_k} - \frac{(\kappa+\mu)\alpha_{k-1}^2\theta_k}{2}\|x^* - v_k\|^2 \leq S_{k-1} + \left(\frac{\delta_k \kappa}{2\min\{1,\theta_k\}} - \frac{\kappa}{2} \right) \|x_k - y_{k-1}\|^2.$$

Then, by choosing $\theta_k = \delta_k < 1$, the quadratic term on the right disappears and the left-hand side is greater than $\frac{1-\delta_k}{1-\alpha_k} S_k$. Thus,

$$S_k \leq \frac{1-\alpha_k}{1-\delta_k} S_{k-1} \leq \frac{A_k}{\prod_{j=1}^k (1-\delta_j)} S_0,$$

which is sufficient to conclude since $(1-\alpha_k)(f(x_k) - f^*) \leq S_k$. □

The next propositions specialize Theorem 8 for specific choices of sequence $(\delta_k)_{k\geq 0}$ in the strongly and non strongly convex cases.

Proposition 4 (μ-strongly convex case, criterion (C2)).
In Algorithm 4, choose $\alpha_0 = \sqrt{q}$ and

$$\delta_k = \frac{\sqrt{q}}{2-\sqrt{q}}.$$

Then, the sequence of iterates $(x_k)_{k\geq 0}$ satisfies

$$f(x_k) - f^* \leqslant 2\left(1 - \frac{\sqrt{q}}{2}\right)^k (f(x_0) - f^*).$$

Proof. This is a direct application of Theorem 8 by remarking that $\gamma_0 = (1-\sqrt{q})\mu$ and

$$S_0 = (1-\sqrt{q})\left(f(x_0) - f^* + \frac{\mu}{2}\|x^* - x_0\|^2 \right) \leq 2(1-\sqrt{q})(f(x_0) - f^*).$$

And $\alpha_k = \sqrt{q}$ for all $k \geq 0$ leading to

$$\frac{1-\alpha_k}{1-\delta_k} = 1 - \frac{\sqrt{q}}{2}$$

□

Proposition 5 (**Convex case, criterion (C2)**).
When $\mu = 0$, choose $\alpha_0 = 1$ and

$$\delta_k = \frac{1}{(k+1)^2}.$$

Then, Algorithm 4 generates iterates $(x_k)_{k\geq 0}$ such that

$$f(x_k) - f^* \leqslant \frac{4\kappa\|x_0 - x^*\|^2}{(k+1)^2}. \tag{3.21}$$

Proof. This is a direct application of Theorem 8 by remarking that $\gamma_0 = \kappa$, $A_k \leq \frac{4}{(k+2)^2}$ (Lemma 18) and

$$\prod_{i=1}^{k}\left(1 - \frac{1}{(i+1)^2}\right) = \prod_{i=1}^{k}\frac{i(i+2)}{(i+1)^2} = \frac{k+2}{2(k+1)} \geq \frac{1}{2}.$$

□

Remark 1. *In fact, the choice of δ_k can be improved by taking $\delta_k = \frac{1}{(k+1)^{1+\gamma}}$ for any $\gamma > 0$, which comes at the price of a larger constant in (3.21).*

3.4.2 Analysis of Warm Start Strategies for the Inner Loop

In this section, we study the complexity of solving the subproblems with the proposed warm start strategies. The only assumption we make on the optimization method $\mathcal{M}$ is that it enjoys linear convergence when solving a strongly convex problem—meaning, it satisfies either (3.12) or its randomized variant (3.13). Then, the following lemma gives us a relation between the accuracy required to solve the sub-problems and the corresponding complexity.

Lemma 8 (**Accuracy vs. complexity**). *Let us consider a strongly convex objective h and a linearly convergent method $\mathcal{M}$ generating a sequence of iterates $(z_t)_{t\geq 0}$ for minimizing h. Consider the complexity $T(\varepsilon) = \inf\{t \geq 0, h(z_t) - h^* \leq \varepsilon\}$, where $\varepsilon > 0$ is the target accuracy and h^* is the minimum value of h. Then,*

1. *If $\mathcal{M}$ is deterministic and satisfies (3.12), we have*

$$T(\varepsilon) \leq \frac{1}{\tau_{\mathcal{M}}}\log\left(\frac{C_{\mathcal{M}}(h(z_0) - h^*)}{\varepsilon}\right).$$

2. *If $\mathcal{M}$ is randomized and satisfies (3.13), we have*

$$\mathbb{E}[T(\varepsilon)] \leq \frac{1}{\tau_{\mathcal{M}}} \log\left(\frac{2C_{\mathcal{M}}(h(z_0) - h^*)}{\tau_{\mathcal{M}}\varepsilon}\right) + 1$$

The proof of the deterministic case is straightforward and the proof of the randomized case is provided in Appendix B.2.4. From the previous result, a good initialization is essential for fast convergence. More precisely, it suffices to control the initialization $\frac{h(z_0)-h^*}{\varepsilon}$ in order to bound the number of iterations $T(\varepsilon)$. For that purpose, we analyze the quality of various warm start strategies.

Warm Start Strategies for Criterion (C1)

The next proposition characterizes the quality of initialization for (C1).

Proposition 6 (Warm start for criterion (C1)). *Assume that $\mathcal{M}$ is linearly convergent for strongly convex problems with parameter $\tau_{\mathcal{M}}$ according to (3.12), or according to (3.13) in the randomized case. At iteration $k+1$ of Algorithm 4, given the previous iterate x_k in $p^{\varepsilon_k}(y_{k-1})$, we consider the following function*

$$h_{k+1}(z) = f(z) + \frac{\kappa}{2}\|z - y_k\|^2,$$

which we minimize with $\mathcal{M}$, producing a sequence $(z_t)_{t\geq 0}$. Then,

- *when f is smooth, choose $z_0 = x_k + \frac{\kappa}{\kappa+\mu}(y_k - y_{k-1})$;*
- *when $f = f_0 + \psi$ is composite, choose $z_0 = \operatorname{prox}_{\eta\psi}(w_0 - \eta\nabla h_0(w_0))$ with $w_0 = x_k + \frac{\kappa}{\kappa+\mu}(y_k - y_{k-1})$, $\eta = \frac{1}{L+\kappa}$ and $h_0 = f_0 + \frac{\kappa}{2}\|\cdot - y_k\|^2$.*

We also assume that we choose α_0 and $(\varepsilon_k)_{k\geq 0}$ according to Proposition 2 for $\mu > 0$, or Proposition 3 for $\mu = 0$. Then,

1. *if f is μ-strongly convex, $h_{k+1}(z_0) - h^*_{k+1} \leq C\varepsilon_{k+1}$ where,*

$$C = \frac{L+\kappa}{\kappa+\mu}\left(\frac{2}{1-\rho} + \frac{2592(\kappa+\mu)}{(1-\rho)^2(\sqrt{q}-\rho)^2\mu}\right) \quad \text{if } f \text{ is smooth,} \tag{3.22}$$

or

$$C = \frac{L+\kappa}{\kappa+\mu}\left(\frac{2}{1-\rho} + \frac{23328(L+\kappa)}{(1-\rho)^2(\sqrt{q}-\rho)^2\mu}\right) \quad \text{if } f \text{ is composite.} \tag{3.23}$$

2. *if f is convex with bounded level sets, there exists a constant $B > 0$ that only depends on f, x_0 and κ such that*

$$h_{k+1}(z_0) - h^*_{k+1} \leq B. \tag{3.24}$$

Proof. We treat the smooth and composite cases separately.

Smooth and strongly-convex case. When f is smooth, by the gradient Lipschitz assumption,

$$h_{k+1}(z_0) - h_{k+1}^* \leq \frac{(L+\kappa)}{2}\|z_0 - p(y_k)\|^2.$$

Moreover,

$$\begin{aligned}\|z_0 - p(y_k)\|^2 &= \|x_k + \frac{\kappa}{\kappa+\mu}(y_k - y_{k-1}) - p(y_k)\|^2 \\ &= \|x_k - p(y_{k-1}) + \frac{\kappa}{\kappa+\mu}(y_k - y_{k-1}) - (p(y_k) - p(y_{k-1}))\|^2 \\ &\leq 2\|x_k - p(y_{k-1})\|^2 + 2\left\|\frac{\kappa}{\kappa+\mu}(y_k - y_{k-1}) - (p(y_k) - p(y_{k-1}))\right\|^2.\end{aligned}$$

Since x_k is in $p^{\varepsilon_k}(y_{k-1})$, we may control the first quadratic term on the right by noting that

$$\|x_k - p(y_{k-1})\|^2 \leq \frac{2}{\kappa+\mu}(h_k(x_k) - h_k^*) \leq \frac{2\varepsilon_k}{\kappa+\mu}.$$

Moreover, by the firmly non-expansiveness property of the proximal operator, it is possible to show that

$$\left\|\frac{\kappa}{\kappa+\mu}(y_k - y_{k-1}) - (p(y_k) - p(y_{k-1}))\right\|^2 \leq \|y_k - y_{k-1}\|^2.$$

As a consequence,

$$\begin{aligned}h_{k+1}(z_0) - h_{k+1}^* &\leq \frac{(L+\kappa)}{2}\|z_0 - p(y_k)\|^2 \\ &\leq (L+\kappa)(\|x_k - p(y_{k-1})\|^2 + \|y_k - y_{k-1} - (p(y_k) - p(y_{k-1}))\|^2), \qquad (3.25) \\ &\leq 2\frac{L+\kappa}{\mu+\kappa}\varepsilon_k + (L+\kappa)\|y_k - y_{k-1}\|^2,\end{aligned}$$

Then, we need to control the term $\|y_k - y_{k-1}\|^2$. Inspired by the proof of accelerated SDCA of Shalev-Shwartz and Zhang [2016],

$$\begin{aligned}\|y_k - y_{k-1}\| &= \|x_k + \beta_k(x_k - x_{k-1}) - x_{k-1} - \beta_{k-1}(x_{k-1} - x_{k-2})\| \\ &\leqslant (1+\beta_k)\|x_k - x_{k-1}\| + \beta_{k-1}\|x_{k-1} - x_{k-2}\| \\ &\leqslant 3\max\left\{\|x_k - x_{k-1}\|, \|x_{k-1} - x_{k-2}\|\right\},\end{aligned}$$

The last inequality was due to the fact that $\beta_k \leq 1$. In fact,

$$\beta_k^2 = \frac{\left(\alpha_{k-1} - \alpha_{k-1}^2\right)^2}{\left(\alpha_{k-1}^2 + \alpha_k\right)^2} = \frac{\alpha_{k-1}^2 + \alpha_{k-1}^4 - 2\alpha_{k-1}^3}{\alpha_k^2 + 2\alpha_k\alpha_{k-1}^2 + \alpha_{k-1}^4} = \frac{\alpha_{k-1}^2 + \alpha_{k-1}^4 - 2\alpha_{k-1}^3}{\alpha_{k-1}^2 + \alpha_{k-1}^4 + q\alpha_k + \alpha_k\alpha_{k-1}^2} \leq 1,$$

where the last equality uses the relation $\alpha_k^2 + \alpha_k\alpha_{k-1}^2 = \alpha_{k-1}^2 + q\alpha_k$ from (3.10). Then,

$$\|x_k - x_{k-1}\| \leqslant \|x_k - x^*\| + \|x_{k-1} - x^*\|,$$

and by strong convexity of f

$$\frac{\mu}{2}\|x_k - x^*\|^2 \leqslant f(x_k) - f^* \leqslant \frac{36}{(\sqrt{q}-\rho)^2}\varepsilon_{k+1},$$

where the last inequality is obtained from Proposition 2. As a result,

$$\begin{aligned}\|y_k - y_{k-1}\|^2 &\leqslant 9\max\left\{\|x_k - x_{k-1}\|^2, \|x_{k-1} - x_{k-2}\|^2\right\} \\ &\leqslant 36\max\left\{\|x_k - x^*\|^2, \|x_{k-1} - x^*\|^2, \|x_{k-2} - x^*\|^2\right\} \\ &\leqslant \frac{2592\,\varepsilon_{k-1}}{(\sqrt{q}-\rho)^2\mu}.\end{aligned}$$

Since $\varepsilon_{k+1} = (1-\rho)^2\varepsilon_{k-1}$, we may now obtain (3.22) from (3.25) and the previous bound.

Smooth and convex case. When $\mu = 0$, Eq. (3.25) is still valid but we need to control $\|y_k - y_{k-1}\|^2$ in a different way. From Proposition 3, the sequence $(f(x_k))_{k\geq 0}$ is bounded by a constant that only depends on f and x_0; therefore, by the bounded level set assumption, there exists $R > 0$ such that

$$\|x_k - x^*\| \leq R, \quad \text{for all } k \geq 0.$$

Thus, following the same argument as the strongly convex case, we have

$$\|y_k - y_{k-1}\| \leq 36R^2 \quad \text{for all } k \geq 1,$$

and we obtain (3.24) by combining the previous inequality with (3.25).

Composite case. By using the notation of gradient mapping introduced in (3.7), we have $z_0 = [w_0]_\eta$. By following similar steps as in the proof of Lemma 6, the gradient mapping satisfies the following relation

$$h_{k+1}(z_0) - h_{k+1}^* \leq \frac{1}{2(\kappa+\mu)}\left\|\frac{1}{\eta}(w_0 - z_0)\right\|^2,$$

and we need to bound $\|w_0 - z_0\| = \|w_0 - [w_0]_\eta\|$. For that, we introduce

$$[x_k]_\eta = \operatorname{prox}_{\eta\psi}(x_k - \eta(\nabla f_0(x_k) + \kappa(x_k - y_{k-1}))).$$

Then,

$$\|w_0 - [w_0]_\eta\| \leq \|w_0 - x_k\| + \|x_k - [x_k]_\eta\| + \|[x_k]_\eta - [w_0]_\eta\|, \tag{3.26}$$

and we will bound each term on the right. By construction

$$\|w_0 - x_k\| = \frac{\kappa}{\kappa+\mu}\|y_k - y_{k-1}\| \leq \|y_k - y_{k-1}\|.$$

Next, it is possible to show that the gradient mapping satisfies the following relation [see Nesterov, 2013],

$$\frac{1}{2\eta}\|x_k - [x_k]_\eta\|^2 \leq h_k(x_k) - h_k^* \leq \varepsilon_k.$$

And then since $[x_k]_\eta = \operatorname{prox}_{\eta\psi}(x_k - \eta(\nabla f_0(x_k) + \kappa(x_k - y_{k-1})))$ and $[w_0]_\eta = \operatorname{prox}_{\eta\psi}(w_0 - \eta(\nabla f_0(w_0) + \kappa(w_0 - y_k)))$. From the non expansiveness of the proximal operator, we have

$$\begin{aligned}
\|[x_k]_\eta - [w_0]_\eta\| &\leq \|x_k - \eta(\nabla f_0(x_k) + \kappa(x_k - y_{k-1})) - (w_0 - \eta(\nabla f_0(w_0) + \kappa(w_0 - y_k)))\| \\
&\leq \|x_k - \eta(\nabla f_0(x_k) + \kappa(x_k - y_{k-1})) - (w_0 - \eta(\nabla f_0(w_0) + \kappa(w_0 - y_{k-1})))\| \\
&\quad + \eta\kappa\|y_k - y_{k-1}\| \\
&\leq \|x_k - w_0\| + \eta\kappa\|y_k - y_{k-1}\| \\
&\leq 2\|y_k - y_{k-1}\|.
\end{aligned}$$

We have used the fact that $\|x - \eta\nabla h(x) - (y - \eta\nabla h(y))\| \leq \|x - y\|$. By combining the previous inequalities with (3.26), we finally have

$$\|w_0 - [w_0]_\eta\| \leq \sqrt{2\eta\varepsilon_k} + 3\|y_k - y_{k-1}\|.$$

Thus, by using the fact that $(a+b)^2 \leq 2a^2 + 2b^2$ for all a, b,

$$h_{k+1}(z_0) - h_{k+1}^* \leq \frac{L+\kappa}{\kappa+\mu}\left(2\varepsilon_k + 9(L+\kappa)\|y_k - y_{k-1}\|^2\right),$$

and we can obtain (3.23) and (3.24) by upper-bounding $\|y_k - y_{k-1}\|^2$ in a similar way as in the smooth case, both when $\mu > 0$ and $\mu = 0$.

□

Finally, the complexity of the inner loop can be obtained directly by combining the previous proposition with Lemma 8.

Corollary 3 (Inner-loop Complexity for Criterion (C1)). *Consider the setting of Proposition 6; then, the sequence $(z_t)_{t\geq 0}$ minimizing h_{k+1} is such that the complexity $T_{k+1} = \inf\{t \geq 0, h_{k+1}(z_t) - h_{k+1}^* \leq \varepsilon_{k+1}\}$ satisfies*

$$T_{k+1} \leq \frac{1}{\tau_{\mathcal{M}}}\log\left(C_{\mathcal{M}}C\right) \quad \text{if } \mu > 0 \quad \Longrightarrow \quad T_{k+1} = \tilde{O}\left(\frac{1}{\tau_{\mathcal{M}}}\right),$$

where C is the constant defined in (3.22) or in (3.23) for the composite case; and

$$T_{k+1} \leq \frac{1}{\tau_{\mathcal{M}}}\log\left(\frac{9C_{\mathcal{M}}(k+2)^{4+\eta}B}{2(f(x_0) - f^*)}\right) \quad \text{if } \mu = 0 \quad \Longrightarrow \quad T_{k+1} = \tilde{O}\left(\frac{\log(k+2)}{\tau_{\mathcal{M}}}\right),$$

where B is the uniform upper bound in (3.24). Furthermore, when $\mathcal{M}$ is randomized, the expected complexity $\mathbb{E}[T_{k+1}]$ is similar, up to a factor $2/\tau_{\mathcal{M}}$ in the logarithm—see Lemma 8, and we have $\mathbb{E}[T_{k+1}] = \tilde{O}(1/\tau_{\mathcal{M}})$ when $\mu > 0$ and $\mathbb{E}[T_{k+1}] = \tilde{O}(\log(k+2)/\tau_{\mathcal{M}})$. Here, $\tilde{O}(.)$ hides logarithmic dependencies in parameters $\mu, L, \kappa, C_{\mathcal{M}}$, $\tau_{\mathcal{M}}$ and $f(x_0) - f^*$.

Warm Start Strategies for Criterion (C2)

We may now analyze the inner-loop complexity for criterion (C2) leading to upper bounds with smaller constants and simpler proofs. Note also that in the convex case, the bounded level set condition will not be needed, unlike for criterion (C1). To proceed, we start with a simple lemma that gives us a sufficient condition for (C2) to be satisfied.

Lemma 9 (Sufficient condition for criterion (C2)). *If a point z satisfies*

$$h_{k+1}(z) - h^* \leq \frac{\delta_{k+1}\kappa}{8}\|p(y_k) - y_k\|^2,$$

then z is in $g^{\delta_{k+1}}(y_k)$.

Proof.

$$\begin{aligned}
h_{k+1}(z) - h^*_{k+1} &\leq \frac{\delta_{k+1}\kappa}{8}\|p(y_k) - y_k\|^2 \\
&\leq \frac{\delta_{k+1}\kappa}{4}\left(\|p(y_k) - z\|^2 + \|z - y_k\|^2\right) \\
&\leq \frac{\delta_{k+1}\kappa}{4}\left(\frac{2}{\mu+\kappa}(h_{k+1}(z) - h^*_{k+1}) + \|z - y_k\|^2\right) \\
&\leq \frac{1}{2}\left(h_{k+1}(z) - h^*_{k+1}\right) + \frac{\delta_{k+1}\kappa}{4}\|z - y_k\|^2.
\end{aligned}$$

Rearranging the terms gives the desired result. □

With the previous result, we can control the complexity of the inner-loop minimization with Lemma 8 by choosing $\varepsilon = \frac{\delta_{k+1}\kappa}{8}\|p(y_k) - y_k\|^2$. However, to obtain a meaningful upper bound, we need to control the ratio

$$\frac{h_{k+1}(z_0) - h^*_{k+1}}{\varepsilon} = \frac{8(h_{k+1}(z_0) - h^*_{k+1})}{\delta_{k+1}\kappa\|p(y_k) - y_k\|^2}.$$

Proposition 7 (Warm start for criterion (C2)). *Assume that $\mathcal{M}$ is linearly convergent for strongly convex problems with parameter $\tau_{\mathcal{M}}$ according to (3.12), or according to (3.13) in the*

randomized case. At iteration $k+1$ of Algorithm 4, given the previous iterate x_k in $g^{\delta_k}(y_{k-1})$, we consider the following function

$$h_{k+1}(z) = f(z) + \frac{\kappa}{2}\|z - y_k\|^2,$$

which we minimize with $\mathcal{M}$, producing a sequence $(z_t)_{t\geq 0}$. Then,

- *when f is smooth, choose $z_0 = y_k$;*
- *when $f = f_0 + \psi$ is composite, choose $z_0 = \mathrm{prox}_{\eta\psi}(y_k - \eta\nabla f_0(y_k))$ with $\eta = \frac{1}{L+\kappa}$.*

Then,

$$h_{k+1}(z_0) - h_{k+1}^* \leq \frac{L+\kappa}{2}\|p(y_k) - y_k\|^2. \tag{3.27}$$

Proof. When f is smooth, the optimality conditions of $p(y_k)$ yield $\nabla h_{k+1}(p(y_k)) = \nabla f(p(y_k)) + \kappa(p(y_k) - y_k) = 0$. As a result,

$$\begin{aligned} h_{k+1}(z_0) - h_{k+1}^* &= f(y_k) - \left(f(p(y_k)) + \frac{\kappa}{2}\|p(y_k) - y_k\|^2\right) \\ &\leq f(p(y_k)) + \langle \nabla f(p(y_k)), y_k - p(y_k)\rangle + \frac{L}{2}\|y_k - p(y_k)\|^2 \\ &\qquad - \left(f(p(y_k)) + \frac{\kappa}{2}\|p(y_k) - y_k\|^2\right) \\ &= \frac{L+\kappa}{2}\|p(y_k) - y_k\|^2. \end{aligned}$$

When f is composite, we use the inequality in Lemma 2.3 of [Beck and Teboulle, 2009]: for any z,

$$h_{k+1}(z) - h_{k+1}(z_0) \geq \frac{L+\kappa}{2}\|z_0 - y_k\|^2 + (L+\kappa)\langle z_0 - y_k, y_k - z\rangle,$$

Then, we apply this inequality with $z = p(y_k)$, and thus,

$$\begin{aligned} h_{k+1}(z_0) - h_{k+1}^* &\leq -\frac{L+\kappa}{2}\|z_0 - y_k\|^2 - (L+\kappa)\langle z_0 - y_k, y_k - p(y_k)\rangle \\ &\leq \frac{L+\kappa}{2}\|p(y_k) - y_k\|^2. \end{aligned}$$

□

We are now in shape to derive a complexity bound for criterion (C2), which is obtained by combining directly Lemma 8 with the value $\varepsilon = \frac{\delta_{k+1}\kappa}{8}\|p(y_k) - y_k\|^2$, Lemma 9, and the previous proposition.

Corollary 4 (**Inner-loop Complexity for Criterion (C2)**). *Consider the setting of Proposition 7 when $\mathcal{M}$ is deterministic; assume further that α_0 and $(\delta_k)_{k\geq 0}$ are chosen according to Proposition 4 for $\mu > 0$, or Proposition 5 for $\mu = 0$.*

Then, the sequence $(z_t)_{t\geq 0}$ is such that the complexity $T_{k+1} = \inf\{t \geq 0, z_t \in g^{\delta_{k+1}}(y_k)\}$ satisfies

$$T_{k+1} \leq \frac{1}{\tau_{\mathcal{M}}} \log\left(4C_{\mathcal{M}} \frac{(L+\kappa)}{\kappa} \frac{2-\sqrt{q}}{\sqrt{q}}\right) \quad \text{when } \mu > 0,$$

and

$$T_{k+1} \leq \frac{1}{\tau_{\mathcal{M}}} \log\left(4C_{\mathcal{M}} \frac{(L+\kappa)}{\kappa}(k+2)^2\right) \quad \text{when } \mu = 0.$$

When $\mathcal{M}$ is randomized, the expected complexity is similar, up to a factor $2/\tau_{\mathcal{M}}$ in the logarithm—see Lemma 8, and we have $\mathbb{E}[T_{k+1}] = \tilde{O}(1/\tau_{\mathcal{M}})$ when $\mu > 0$ and $\mathbb{E}[T_{k+1}] = \tilde{O}(\log(k+2)/\tau_{\mathcal{M}})$.

The inner-loop complexity is asymptotically similar with criterion (C2) as with criterion (C1), but the constants are significantly better.

3.4.3 Global Complexity Analysis

In this section, we combine the previous outer-loop and inner-loop convergence results to derive a global complexity bound. We treat here the strongly convex ($\mu > 0$) and convex ($\mu = 0$) cases separately.

Strongly Convex Case

When the problem is strongly convex, we remark that the subproblems are solved in a constant number of iterations $T_k = T = \tilde{O}\left(\frac{1}{\tau_{\mathcal{M}}}\right)$ for both criteria (C1) and (C2). This means that the iterate x_k in Algorithm 4 is obtained after $s = kT$ iterations of the method $\mathcal{M}$. Thus, the true convergence rate of Catalyst applied to $\mathcal{M}$ is of the form

$$f_s - f^* = f\left(x_{\frac{s}{T}}\right) - f^* \leq C'(1-\rho)^{\frac{s}{T}}(f(x_0) - f^*) \leq C'\left(1 - \frac{\rho}{T}\right)^s (f(x_0) - f^*), \tag{3.28}$$

where $f_s = f(x_k)$ is the function value after s iterations of $\mathcal{M}$ and $\rho = \sqrt{q}/2$ for criterion (C2) and ρ may be chosen to be $0.9\sqrt{q}$ for criterion (C1). Then, choosing κ consists of maximizing the rate of convergence (3.28). In other words, we want to maximize $\sqrt{q}/T = \tilde{O}(\sqrt{q}\tau_{\mathcal{M}})$. Since $q = \frac{\mu}{\mu+\kappa}$, this naturally lead to the maximization of $\tau_{\mathcal{M}}/\sqrt{\mu+\kappa}$. As we will discuss later, this choice recovers classical accelerated rates of convergence. We now state more formally the global convergence result in terms of complexity.

Proposition 8 (**Global Complexity for strongly convex objectives**). *When f is μ-strongly convex and all parameters are chosen according to Propositions 2 and 6 when using*

criterion (C1), or Propositions 4 and 7 for (C2), then Algorithm 4 finds a solution $\hat{x}$ such that $f(\hat{x}) - f^* \leq \varepsilon$ in at most $N_{\mathcal{M}}$ iterations of a deterministic method $\mathcal{M}$ with

1. *when criterion (C1) is used,*

$$N_{\mathcal{M}} \leq \frac{1}{\tau_{\mathcal{M}}\rho} \log\left(C_{\mathcal{M}} C\right) \cdot \log\left(\frac{8(f(x_0) - f^*)}{(\sqrt{q} - \rho)^2 \varepsilon}\right) = \tilde{O}\left(\frac{1}{\tau_{\mathcal{M}}\sqrt{q}} \log\left(\frac{1}{\varepsilon}\right)\right),$$

   where $\rho = 0.9\sqrt{q}$ and C is the constant defined in (3.22) or (3.23) for the composite case;

2. *when criterion (C2) is used,*

$$N_{\mathcal{M}} \leq \frac{2}{\tau_{\mathcal{M}}\sqrt{q}} \log\left(4 C_{\mathcal{M}} \frac{L + \kappa}{\kappa} \frac{2 - \sqrt{q}}{\sqrt{q}}\right) \cdot \log\left(\frac{2(f(x_0) - f^*)}{\varepsilon}\right) = \tilde{O}\left(\frac{1}{\tau_{\mathcal{M}}\sqrt{q}} \log\left(\frac{1}{\varepsilon}\right)\right).$$

Note that similar results hold in terms of expected number of iterations when the method $\mathcal{M}$ is randomized (see the end of Proposition 6).

Proof. Let K be the number of iterations of the outer-loop algorithm required to obtain an ε-accurate solution. From Proposition 2, using (C1) criterion yields

$$K \leq \frac{1}{\rho} \log\left(\frac{8(f(x_0) - f^*)}{(\sqrt{q} - \rho)^2 \varepsilon}\right).$$

From Proposition 4, using (C2) criterion yields

$$K \leq \frac{2}{\sqrt{q}} \log\left(\frac{2(f(x_0) - f^*)}{\varepsilon}\right).$$

Then since the number of runs of $\mathcal{M}$ is constant for any inner loop, the total number $N_{\mathcal{M}}$ is given by KT where T is respectively given by Corollaries 3 and 4. □

Convex, but not Strongly Convex Case

When $\mu = 0$, the number of iterations for solving each subproblems grows logarithmically, which means that the iterate x_k in Algorithm 4 is obtained after $s \leq kT \log(k+2)$ iterations of the method $\mathcal{M}$, where T is a constant. By using the global iteration counter $s = kT \log(k+2)$, we finally have

$$f_s - f^* \leq C' \frac{\log^2(s)}{s^2} \left(f(x_0) - f^* + \frac{\kappa}{2}\|x_0 - x^*\|^2\right). \tag{3.29}$$

This rate is *near-optimal*, up to a logarithmic factor, when compared to the optimal rate $O(1/s^2)$. This may be the price to pay for using a generic acceleration scheme. As before, we formally detail the global complexity bound for convex in the next proposition.

Proposition 9 (Global complexity for convex objectives). *When f is convex and all parameters are chosen according to Propositions 3 and 6 when using criterion (C1), or Propositions 5 and 7 for criterion (C2), then Algorithm 4 finds a solution $\hat{x}$ such that $f(\hat{x}) - f^* \leq \varepsilon$ in at most $N_{\mathcal{M}}$ iterations of a deterministic method $\mathcal{M}$ with*

1. *when criterion (C1) is applied*

$$N_{\mathcal{M}} \leq \frac{1}{\tau_{\mathcal{M}}} K \log\left(\frac{9 C_{\mathcal{M}} B K^{4+\gamma}}{2(f(x_0) - f^*)}\right) = \tilde{O}\left(\frac{1}{\tau_{\mathcal{M}}}\sqrt{\frac{\kappa}{\varepsilon}} \log\left(\frac{1}{\varepsilon}\right)\right),$$

where,

$$K_\varepsilon = \sqrt{\frac{8\left(\frac{\kappa}{2}\|x_0 - x^*\|^2 + \frac{4}{\gamma^2}(f(x_0) - f^*)\right)}{\varepsilon}};$$

2. *when criterion (C2) is applied,*

$$\begin{aligned} N_{\mathcal{M}} &\leq \frac{1}{\tau_{\mathcal{M}}}\sqrt{\frac{4\kappa\|x_0 - x^*\|^2}{\varepsilon}} \log\left(\frac{16 C_{\mathcal{M}}(L+\kappa)\|x_0 - x^*\|^2}{\varepsilon}\right) \\ &= \tilde{O}\left(\frac{1}{\tau_{\mathcal{M}}}\sqrt{\frac{\kappa}{\varepsilon}} \log\left(\frac{1}{\varepsilon}\right)\right). \end{aligned}$$

Note that similar results hold in terms of expected number of iterations when the method $\mathcal{M}$ is randomized (see the end of Proposition 7).

Proof. Let K denote the number of outer-loop iterations required to achieve an ε-accurate solution. From Proposition 3, when (C1) is applied, we have

$$K \leq \sqrt{\frac{8\left(\frac{\kappa}{2}\|x_0 - x^*\|^2 + \frac{4}{\gamma^2}(f(x_0) - f^*)\right)}{\varepsilon}}.$$

From Proposition 5, when (C2) is applied, we have

$$K \leq \sqrt{\frac{4\kappa\|x_0 - x^*\|^2}{\varepsilon}}.$$

Since the number of runs in the inner loop is increasing, we have

$$N_{\mathcal{M}} = \sum_{i=1}^{K} T_i \leq K T_K.$$

Respectively apply T_K obtained from Corollary 3 and Corollary 4 gives the result. □

Theoretical foundations of the choice of κ. The parameter κ plays an important rule in the global complexity result. The linear convergence parameter $\tau_{\mathcal{M}}$ depends typically on κ since it controls the strong convexity parameter of the subproblems. The natural way to choose κ is to minimize the global complexity given by Proposition 8 and Proposition 9, which leads to the following rule

$$\textbf{Choose } \boldsymbol{\kappa} \textbf{ to maximize } \frac{\tau_{\mathcal{M}}}{\sqrt{\mu+\kappa}},$$

where $\mu = 0$ when the problem is convex but not strongly convex. We now illustrate two examples when applying Catalyst to the classical gradient descent method and to the incremental approach SVRG.

Gradient descent. When $\mathcal{M}$ is the gradient descent method, we have

$$\tau_{\mathcal{M}} = \frac{\mu+\kappa}{L+\kappa}.$$

Maximizing the ratio $\frac{\tau_{\mathcal{M}}}{\sqrt{\mu+\kappa}}$ gives

$$\kappa = L - 2\mu, \qquad \text{when } L > 2\mu.$$

Consequently, the complexity in terms of gradient evaluations for minimizing the finite sum 3.2, where each iteration of $\mathcal{M}$ cost n gradients, is given by

$$N_{\mathcal{M}} = \begin{cases} \tilde{O}\left(n\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right) & \text{when } \mu > 0; \\ \tilde{O}\left(n\sqrt{\frac{L}{\varepsilon}}\log\left(\frac{1}{\varepsilon}\right)\right) & \text{when } \mu = 0. \end{cases}$$

More precisely, when (C2) is applied

$$N_{\mathcal{M}} \leq \begin{cases} \sqrt{\frac{2(L-\mu)}{\mu}}\log\left(\frac{2(f(x_0)-f^*)}{\varepsilon}\right)\log\left(16\sqrt{\frac{L-\mu}{\mu}}\right) & \text{when } \mu > 0; \\ 4\sqrt{\frac{L\|x_0-x^*\|^2}{\varepsilon}}\log\left(\frac{32\|x_0-x^*\|^2}{\varepsilon}\right) & \text{when } \mu = 0. \end{cases}$$

These rates are near-optimal up to logarithmic constants according to the first-order lower bound [Nemirovskii et al., 1983, Nesterov, 2004].

SVRG. For SVRG [Xiao and Zhang, 2014] applied to the same finite-sum objective,

$$\tau_{\mathcal{M}} = \frac{1}{600\left(n + \frac{L+\kappa}{\mu+\kappa}\right)}.$$[5]

[5]The coefficient 600 corresponds to the choice $\theta = 0.1$ and $m = 100\frac{L}{\mu}$ in Xiao and Zhang [2014].

Thus, maximizing the corresponding ratio gives

$$\kappa = \frac{L-\mu}{n+1} - \mu, \qquad \text{when } L > (n+2)\mu.$$

Consequently, the resulting global complexity, here in terms of expected number of gradient evaluations, is given by

$$\mathbb{E}[N_{\mathcal{M}}] = \begin{cases} \tilde{O}\left(\sqrt{n\frac{L}{\mu}}\log\left(\frac{1}{\varepsilon}\right)\right) & \text{when } \mu > 0 \text{ and } L > (n+2)\mu; \\ \tilde{O}\left(\sqrt{\frac{nL}{\varepsilon}}\log\left(\frac{1}{\varepsilon}\right)\right) & \text{when } \mu = 0. \end{cases}$$

Note that we treat here only the ill-conditioned case $L > (n+2)\mu$ to simplify. More precisely, when (C2) is applied

$$\mathbb{E}[N_{\mathcal{M}}] \leq \begin{cases} 4800\sqrt{n\frac{L}{\mu}}\log\left(\frac{2(f(x_0)-f^*)}{\varepsilon}\right)\log\left(76800n\sqrt{n\frac{L}{\mu}}\right) & \text{when } \mu > 0 \text{ and } L > (n+2)\mu; \\ 9600\sqrt{\frac{nL}{\varepsilon}}\log\left(153600\frac{nL\|x_0-x^*\|^2}{\varepsilon}\right) & \text{when } \mu = 0. \end{cases}$$

We end up with large constant coefficients since we use a pessimistic convergence rate of SVRG algorithm, which is purely theoretical. This rate is near-optimal up to logarithmic factors according to the first-order lower bound [Woodworth and Srebro, 2016, Arjevani and Shamir, 2016]. We remark that applying Catalyst to other incremental algorithms such as SAG/SAGA [Schmidt et al., 2017, Defazio et al., 2014a] or dual-type algorithm Finito/MISO [Defazio et al., 2014b, Mairal, 2015] or SDCA [Shalev-Shwartz and Zhang, 2012] yields similar convergence rates, which is different only in the constant. Thus, we successfully accelerate a large class of first-order algorithms, see Table 3.1 for a summary.

Practical Aspects of the Theoretical Analysis

So far, we have not discussed the fixed budget criterion mentioned in Section 3.3.1. The idea is quite natural and simple to implement: we predefine the number of iterations to run for solving each subproblems and stop worrying about the stopping condition. For example, when $\mu > 0$ and $\mathcal{M}$ is deterministic, we can simply run $T_{\mathcal{M}}$ iterations of $\mathcal{M}$ for each subproblem where $T_{\mathcal{M}}$ is greater than the value given by Corollaries 3 or 4, then the criterions (C1) and (C2) are guaranteed to be satisfied. Unfortunately, the theoretical bound of $T_{\mathcal{M}}$ is relatively poor and does not lead to a practical strategy. On the other hand, using a more aggressive strategy such as $T_{\mathcal{M}} = n$ for incremental algorithms, meaning one pass over the data, seems to provide outstanding results, as shown in the experimental part of this chapter.

Finally, one could argue that choosing κ according to a worst-case convergence analysis is not necessarily a good choice. In particular, the convergence rate of the method $\mathcal{M}$, driven by the parameter $\tau_{\mathcal{M}}$ is probably often under estimated in the first place. This suggests that using a smaller value for κ than the one we have advocated earlier is a good thing. In practice, we have

observed that indeed Catalyst is often robust to smaller values of κ than the theoretical one, but we have also observed that the theoretical value performs reasonably well (see experimental section).

3.5 Catalyst for MISO/Finito/SDCA

In this section, we present the application of Catalyst to MISO/Finito [Mairal, 2015, Defazio et al., 2014b], which may be seen as a variant of SDCA [Shalev-Shwartz and Zhang, 2016]. The reason why these algorithms require a specific treatment is due to the fact that their linear convergence rates are given in a different form than (3.12); specifically, Theorem 4.1 of Lin et al. [2015] tells us that MISO produces a sequence of iterates $(z_t)_{t\geq 0}$ for minimizing the auxiliary objective $h(z) = f(z) + \frac{\kappa}{2}\|z - y\|^2$ such that

$$\mathbb{E}[h(z_t)] - h^* \leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^{t+1}(h^* - d_0(z_0)),$$

where d_0 is a lower-bound of h defined as the sum of a simple quadratic function and the composite regularization ψ. More precisely, these algorithms produce a sequence $(d_t)_{t\geq 0}$ of such lower-bounds, and the iterate z_t is obtained by minimizing d_t in closed form. In particular, z_t is obtained from taking a proximal step at a well chosen point w_t, providing the following expression,

$$z_t = \operatorname{prox}_{\psi/(\kappa+\mu)}(w_t).$$

Then, linear convergence is achieved for the duality gap

$$\mathbb{E}[h(z_t) - h^*] \leq \mathbb{E}[h(z_t) - d_t(z_t)] \leq C_{\mathcal{M}}(1 - \tau_{\mathcal{M}})^t(h^* - d_0(z_0)).$$

Indeed, the quantity $h(z_t) - d_t(z_t)$ is a natural upper-bound on $h(z_t) - h^*$, which is simple to compute, and which can be naturally used for checking the criterions (C1) and (C2). Consequently, the expected complexity of solving a given problem is slightly different compared to Lemma 8.

Lemma 10 (Accuracy vs. complexity). *Let us consider a strongly convex objective h. Finito/MISO/SDCA generate a sequence of iterates $(z_t)_{t\geq 0}$. Consider the complexity $T(\varepsilon) = \inf\{t \geq 0, h(z_t) - d_0(z_0) \leq \varepsilon\}$, where $\varepsilon > 0$ is the target accuracy and h^* is the minimum value of h. Then,*

$$\mathbb{E}[T(\varepsilon)] \leq \frac{1}{\tau_{\mathcal{M}}} \log\left(\frac{2C_{\mathcal{M}}(h^* - d_0(z_0))}{\tau_{\mathcal{M}}\varepsilon}\right) + 1,$$

where d_0 is a lower bound of f constructed by the algorithm.

For the convergence analysis, the outer-loop complexity does not change as long as the algorithm finds approximate proximal points satisfying criterions (C1) and (C2). It is then sufficient to control the inner loop complexity. As we can see, we now need to bound the dual gap $h^* - d_0(z_0)$ instead of the primal gap $h(z_0) - h^*$, leading to slightly different warm start strategies. Here, we show how to warm start MISO/Finito.

Proposition 10 (**Warm start for criterion (C1)**). *Consider applying Catalyst with the same parameter choices as in Proposition 6 to MISO/Finito. At iteration $k+1$ of Algorithm 4, assume that we are given the previous iterate x_k in $p^{\varepsilon_k}(y_{k-1})$, the corresponding dual function $d(x)$ and its prox-center w_k satisfying $x_k = \mathrm{prox}_{\psi/(\kappa+\mu)}(w_k)$. Then, initialize the sequence $(z_t)_{t\geq 0}$ for minimizing $h_{k+1} = f + \frac{\kappa}{2}\|\cdot - y_k\|^2$ with,*

$$z_0 = \mathrm{prox}_{\psi/(\kappa+\mu)}\left(w_k + \frac{\kappa}{\kappa+\mu}(y_k - y_{k-1})\right),$$

and initialize the dual function as

$$d_0(x) = d(x) + \frac{\kappa}{2}\|x - y_k\|^2 - \frac{\kappa}{2}\|x - y_{k-1}\|^2.$$

Then,

1. *when f is μ-strongly convex, we have $h^*_{k+1} - d_0(z_0) \leq C\varepsilon_{k+1}$ with the same constant as in (3.22) and (3.23), where d_0 is the dual function corresponding to z_0;*
2. *when f is convex with bounded level sets, there exists a constant $B > 0$ identical to the one of (3.24) such that*
$$h^*_{k+1} - d_0(z_0) \leq B.$$

Proof. The proof is given in Lemma D.5 of [Lin et al., 2015], which gives

$$h^*_{k+1} - d_0(z_0) \leq \varepsilon_k + \frac{\kappa^2}{2(\kappa+\mu)}\|y_k - y_{k-1}\|^2.$$

This term is smaller than the quantity derived from (3.25), leading to the same upper bound. □

Proposition 11 (**W**arm start for criterion (C2)). *Consider applying Catalyst with the same parameter choices as in Proposition 7 to MISO/Finito. At iteration $k+1$ of Algorithm 4, we assume that we are given the previous iterate x_k in $g^{\delta_k}(y_{k-1})$ and the corresponding dual function $d(x)$. Then, initialize the sequence $(z_t)_{t\geq 0}$ for minimizing $h_{k+1} = f + \frac{\kappa}{2}\|\cdot - y_k\|^2$ by*

$$z_0 = \mathrm{prox}_{\psi/(\kappa+\mu)}\left(y_k - \frac{1}{\kappa+\mu}\nabla f_0(y_k)\right),$$

where $f = f_0 + \psi$ and f_0 is the smooth part of f, and set the dual function d_0 by

$$d_0(x) = f_0(y_k) + \langle \nabla f_0(y_k), x - y_k\rangle + \frac{\kappa+\mu}{2}\|x - y_k\|^2 + \psi(x).$$

Then,

$$h^*_{k+1} - d_0(z_0) \leq \frac{(L+\kappa)^2}{2(\mu+\kappa)}\|p(y_k) - y_k\|^2. \tag{3.30}$$

Proof. Since $p(y_k)$ is the minimum of h_{k+1}, the optimality condition provides

$$-\nabla f_0(p(y_k)) - \kappa(p(y_k) - y_k) \in \partial\psi(p(y_k)).$$

Thus, by convexity,

$$\psi(p(y_k)) + \langle -\nabla f_0(p(y_k)) - \kappa(p(y_k) - y_k), z_0 - p(y_k)\rangle \leq \psi(z_0),$$
$$f_0(p(y_k)) + \frac{\kappa}{2}\|p(y_k) - y_k\|^2 + \langle \nabla f_0(p(y_k)) + \kappa(p(y_k) - y_k), y_k - p(y_k)\rangle \leq f_0(y_k).$$

Summing up gives

$$h^*_{k+1} \leq f_0(y_k) + \psi(z_0) + \langle \nabla f_0(p(y_k)) + \kappa(p(y_k) - y_k), z_0 - y_k\rangle.$$

As a result,

$$\begin{aligned} h^*_{k+1} - d_0(z_0) &\leq f_0(y_k) + \psi(z_0) + \langle \nabla f_0(p(y_k)) + \kappa(p(y_k) - y_k), z_0 - y_k\rangle - d_0(z_0) \\ &= \langle \nabla f_0(p(y_k)) + \kappa(p(y_k) - y_k) - \nabla f_0(y_k), z_0 - y_k\rangle - \frac{\kappa+\mu}{2}\|z_0 - y_k\|^2 \\ &\leq \frac{1}{2(\kappa+\mu)}\|\nabla \underbrace{f_0(p(y_k)) - \nabla f_0(y_k)}_{\|\cdot\| \leq L\|p(y_k) - y_k\|} + \kappa(p(y_k) - y_k)\|^2 \\ &\leq \frac{(L+\kappa)^2}{2(\mu+\kappa)}\|p(y_k) - y_k\|^2. \end{aligned}$$

□

The bound obtained from (3.30) is similar to the one form Proposition 7, and differs only in the constant factor. Thus, the inner loop complexity in Section 3.4.2 still holds for MISO/Finito up to a constant factor. As a consequence, the global complexity of MISO/Finito applied to Catalyst is similar to one obtained by SVRG, yielding an acceleration for ill-conditioned problems.

3.6 Experimental study

In this section, we conduct various experiments to study the effect of the Catalyst acceleration and its different variants, showing in particular how to accelerate SVRG, SAGA and MISO.

3.6.1 Datasets, formulations, and metric

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

name	covtype	alpha	real-sim	rcv1
n	581 012	250 000	72 309	781 265
d	54	500	20 958	47 152

Formulations. 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 $(b_i, a_i)_{i=1}^n$ of n data points, where the b_i's are scalars in $\{-1, +1\}$ and the a_i are feature vectors in $\mathbb{R}^d$. Then, the goal is to fit a linear model x in $\mathbb{R}^p$ 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 [Zou and Hastie, 2005] leading to strongly-convex optimization problem.

Each feature vector a_i is normalized, and a natural upper-bound on the Lipschitz constant L of the un-regularized objective can be easily obtained with $L_{\text{logistic}} = 1/4$ and $L_{\text{lasso}} = 1$. The regularization parameter μ and λ are choosing in the following way:

- For **Logistic Regression**, we set $\mu = 0.01/n$ which corresponds to relatively ill-conditioned problems.
- For **Elastic-Net**, we set $\mu = 0.01/n$ as in logistic regression and add a small l_1-regularization penalty with $\lambda = 1/n$ that produces sparse solutions.
- 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, which leads to $\lambda = 10/n$ or $100/n$.

Note that for the strongly convex problems, the regularization parameter μ yields a lower bound on the strong convexity parameter of the problem.

Metric used. In this chapter, and following previous work about incremental methods [Schmidt et al., 2017], we plot objective values as a function of the number of gradients evaluated during optimization, which appears to be the computational bottleneck of all previously mentioned algorithms. Since no metric is perfect for comparing algorithms speed, we shall make the two following remarks, such that the reader can interpret our results and the limitations of our study with no difficulty.

- Ideally, CPU-time would be a gold standard but CPU time is implementation-dependent and hardware-dependent. Thus, we have chosen the number of gradient computations as a proxy for CPU time that does not suffer from unwanted externalities.
- We have chosen to count only gradients computed with random data access. Thus, computing n times a gradient f_i by picking each time one function at random counts as "n gradients", whereas we ignore the cost of computing a full gradient $\nabla(1/n)\sum_{i=1}^n f_i$, where the f_i's can be accessed in sequential order. Similarly, we ignore the cost of computing the function value $f(x) = (1/n)\sum_{i=1}^n f_i(x)$, which is typically performed every pass on the data when computing a duality gap. While this assumption may be inappropriate in some contexts, the cost of sequential gradient computations was indeed insignificant in our experiments, where (i) datasets fit into memory; (ii) computing full gradients was done in C++ by calling BLAS2 functions exploiting multiple cores.

3.6.2 Choice of hyper-parameters and variants

Before presenting the numerical results, we discuss the choice of default parameters used in the experiments as well as different variants.

Choice of method $\mathcal{M}$. We consider the acceleration of incremental algorithms which are able to adapt to the problem structure we consider: large sum of functions and composite regularization.

- The proximal SVRG algorithm [Xiao and Zhang, 2014] with stepsize $\eta = 1/L$.
- SAGA algorithm [Defazio et al., 2014a] with stepsize $\eta = 1/3L$.
- The proximal MISO algorithm that we developed in Section 2.3.

Choice of regularization parameter κ. As suggested by the theoretical analysis, we take κ to minimize the global complexity. Leading to the choice

$$\textbf{For SVRG/SAGA/MISO, } \boldsymbol{\kappa = \frac{L-\mu}{n+1} - \mu.}$$

Stopping criteria for the inner loop. The choice of the accuracies are driven from the theoretical analysis. Here, we specify it for the problems we consider:

- **Stopping criterion (C1).** Stop when $h_k(z_t) - h_k^* \leq \varepsilon_k$, where

$$\varepsilon_k = \begin{cases} \frac{2}{9}(1-\rho)^k f(x_0)^{6} \text{ with } \rho = 0.9\sqrt{\frac{\mu}{\mu+\kappa}} & \text{when } \mu > 0; \\ \frac{2f(x_0)}{9(k+1)^{4.1}} & \text{when } \mu = 0. \end{cases}$$

  The duality gap $h(w_t) - h^*$ can be estimated either by evaluating the Fenchel conjugate or by computing the squared norm of the gradient.

- **Stopping criterion (C2).** Stop when $h_k(z_t) - h_k^* \leq \delta_k \cdot \frac{\kappa}{2}\|z_t - y_{k-1}\|^2$, where

$$\delta_k = \begin{cases} \frac{\sqrt{q}}{2-\sqrt{q}} \text{ with } q = \frac{\mu}{\mu+\kappa} & \text{when } \mu > 0; \\ \frac{1}{(k+1)^2} & \text{when } \mu = 0. \end{cases}$$

- **Stopping criterion (C3) .** Perform exactly one pass over the data in the inner loop without checking any stopping criteria.[7]

Warm start for the inner loop. This is an important point to achieve acceleration which was not highlighted in the conference paper [Lin et al., 2015]. At iteration k, we consider the minimization of

$$h_k(z) = f_0(z) + \frac{\kappa}{2}\|z - y_{k-1}\|^2 + \psi(z).$$

Let x_{k-1} be the approximate minimizer of h_{k-1}, obtained from the last iteration.

- **Initialization for (C1).** Let us define $\eta = \frac{1}{L+\kappa}$, then initialize at

$$z_0^{C1} = \begin{cases} w_0 \triangleq x_{k-1} + \frac{\kappa}{\kappa+\mu}(y_{k-1} - y_{k-2}) & \text{if } \psi = 0; \\ \text{prox}_{\eta\psi}(w_0 - \eta g) \text{ with } g = \nabla f_0(w_0) + \kappa(w_0 - y_{k-1}) & \text{otherwise.} \end{cases}$$

- **Initialization for (C2).** Intialize at

$$z_0^{C2} = \begin{cases} y_{k-1} & \text{if } \psi = 0; \\ \text{prox}_{\eta\psi}(y_{k-1} - \eta\nabla f_0(y_{k-1})) & \text{otherwise.} \end{cases}$$

[7] This stopping criterion is heuristic since one pass may not be enough to achieve the required accuracy. What we have shown is that with a large enough $T_{\mathcal{M}}$, then the convergence will be guaranteed. Here we take heuristically $T_{\mathcal{M}}$ as one pass.

- **Initialization for (C3).** Take the best initial point among x_{k-1} and z_0^{C1}

$$z_0^{C3} \quad \text{such that} \quad h_k(z_0^{C3}) = \min\{h_k(x_{k-1}), h_k(z_0^{C1})\}.$$

- **Initialization for (C1*).** Use the strategy (C1) with z_0^{C3}.

The warm start at z_0^{C3} requires to choose the best point between the last iterate x_{k-1} and the point z_0^{C1}. The motivation is that since the one-pass strategy is an aggressive heuristic, the solution of the subproblems may not be as accurate as the ones obtained with other criterions. Allowing to use the iterate x_{k-1} turned out to be significantly more stable in practice. Then, it is also natural to use a similar strategy for criterion (C1), which we call (C1*) . Using a similar strategy for (C2) turned out not to provide any benefit in practice and is thus omitted from the list here.

3.6.3 Comparison of Stopping Criteria and Warm Start Strategies

First, we evaluate the performance of the previous strategies when applying Catalyst to SVRG, SAGA and MISO. The results are presented in Figures 3.1, 3.2, and 3.3, respectively.

Observations for Catalyst-SVRG. We remark that in most of the cases, the curve of (C3) and (C1*) are superimposed, meaning that one pass through the data is enough for solving the subproblem up to the required accuracy. Moreover, they give the best performance among all criterions. Regarding the logistic regression problem, the acceleration is significant (even huge for the covtype dataset) except for alpha, where only (C3) and (C1*) do not degrade significantly the performance. For sparse problems, the effect of acceleration is more mitigated, with 4 cases out of 8 exhibiting important acceleration and 4 cases no acceleration. As before, (C3) and (C1*) are the only strategies that never degrade performance.

One reason explaining why acceleration is not systematic may be the ability of incremental methods to adapt to the unknown strong convexity parameter $\mu' \geq \mu$ hidden in the objective's loss, or local strong convexity near the solution. When $\mu'/L \geq 1/n$, we indeed obtain a well-conditioned regime where acceleration should not occur theoretically since the complexity $O(n \log(1/\varepsilon))$ of the unaccelerated method is already optimal. For sparse problems, conditioning of the problem with respect to the linear subspace where the solution lies might also play a role, even though our analysis does not study this aspect. Therefore, this experiment suggests that adaptivity to unknown strong convexity is of high interest for incremental optimization.

Observations for Catalyst-SAGA. Our conclusions with SAGA are almost the same as with SVRG. However, in a few cases, we also notice that criterion C1 lacks stability, or at least exhibits some oscillations, which may suggest that SAGA has a larger variance compared to SVRG. The difference in the performance of (C1) and (C1*) can be huge, while they differ from each other only by the warm start strategy. Thus, *choosing a good initial point for solving the sub-problems is a key for obtaining acceleration in practice.*

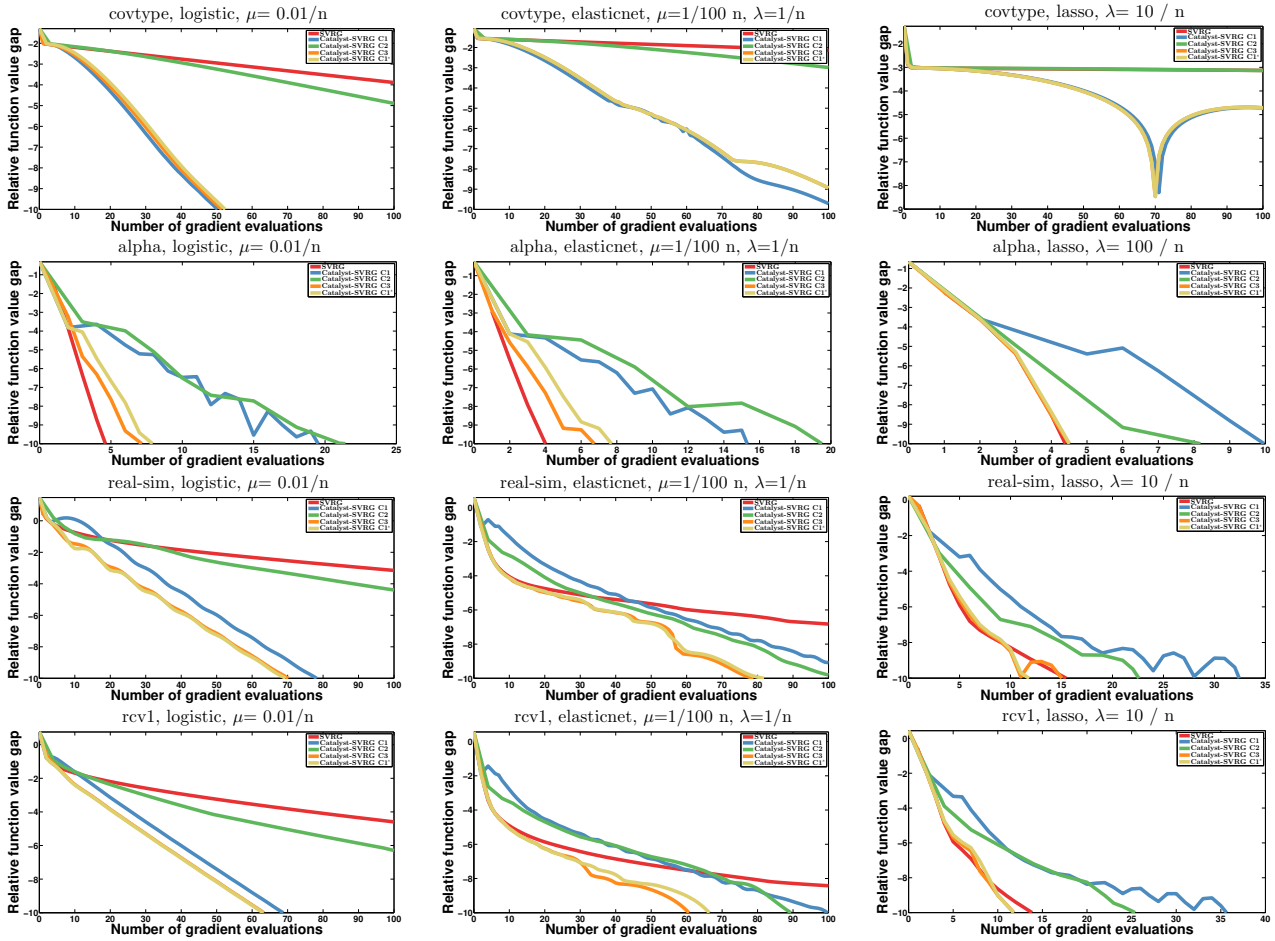

Figure 3.1 – Experimental study of different stopping criterions for Catalyst-SVRG. We plot the value $f(x_k)/f^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value f^* is estimated with a duality gap.

Observations for Catalyst-MISO. The warm start strategy of MISO is different from primal algorithms because parameters for the dual function need to be specified. The most natural way for warm starting the dual functions is to set

$$D_0^{k+1}(x) = D_k(x) + \frac{\kappa}{2}\|x - y_k\|^2 - \frac{\kappa}{2}\|x - y_{k-1}\|^2,$$

where D_k is the last dual function of the previous subproblem h_k. This gives the warm start

$$z_0 = \text{prox}\left(x_k + \frac{\kappa}{\kappa + \mu}(y_k - y_{k-1})\right).$$

For other choices of z_0, the dual function needs to be recomputed from scratch, which is computationally expensive and unstable for ill-conditioned problems. Thus, we only present

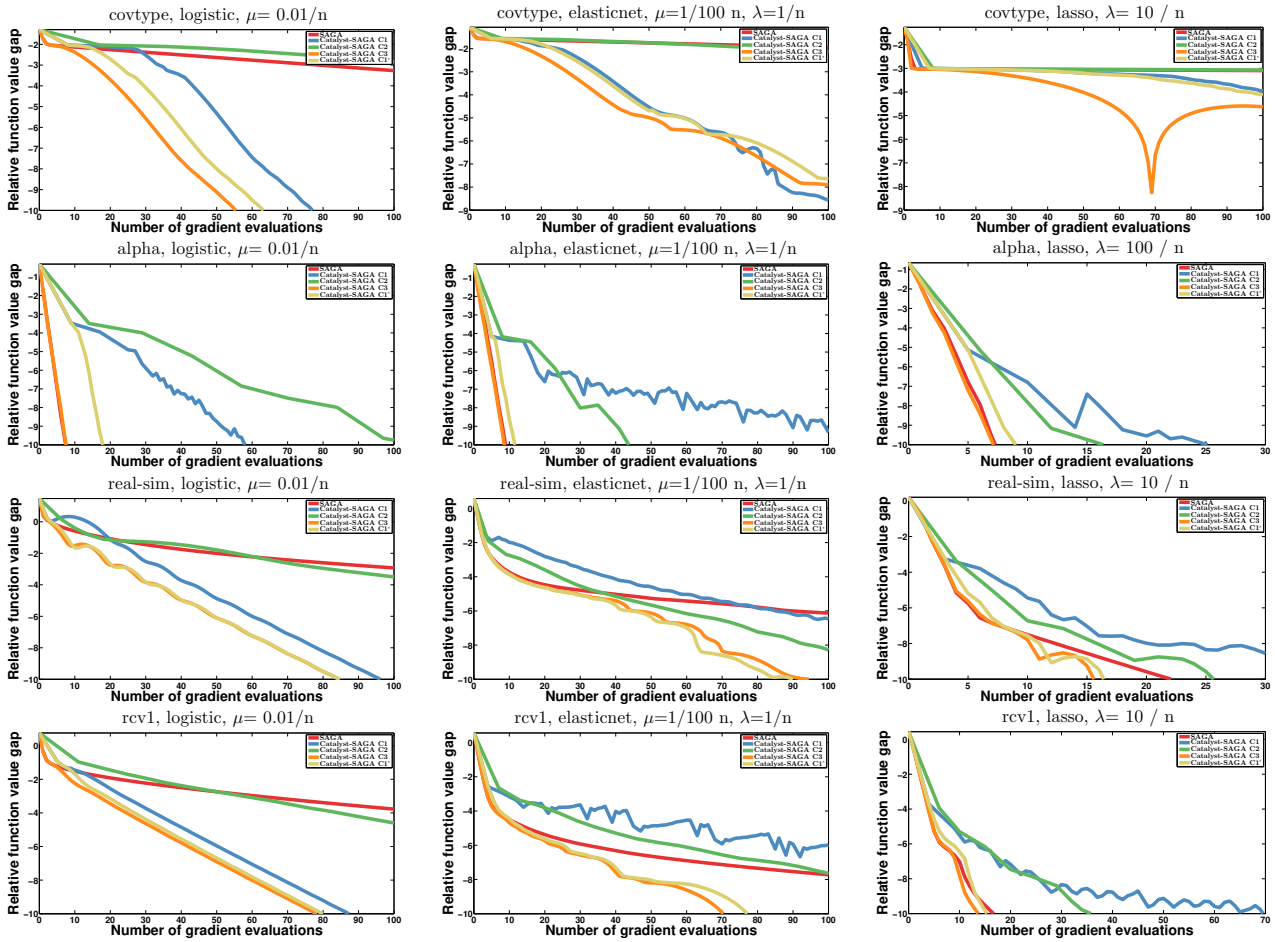

Figure 3.2 – Experimental study of different stopping criterions for Catalyst-SAGA, with a similar setting as in Figure 3.1.

the experimental results with respect to criterion (C1) and the one-pass heuristic (C3) . As we observe, a huge acceleration is obtained in logistic regression and Elastic-net formulations. For Lasso problem, the original Prox-MISO is not defined since the problem is not strongly convex. Thus, in order to make a comparison, we compare with Catalyst-SVRG which shows that the acceleration achieves a similar performance. This confirms the theoretical result that Catalyst applied to incremental algorithms yields a similar convergence rate. Notice also that the original MISO algorithm suffers from numerical instability in this ill-conditioned regime chosen for our experiments. Catalyst not only accelerates MISO, but it also stabilizes it.

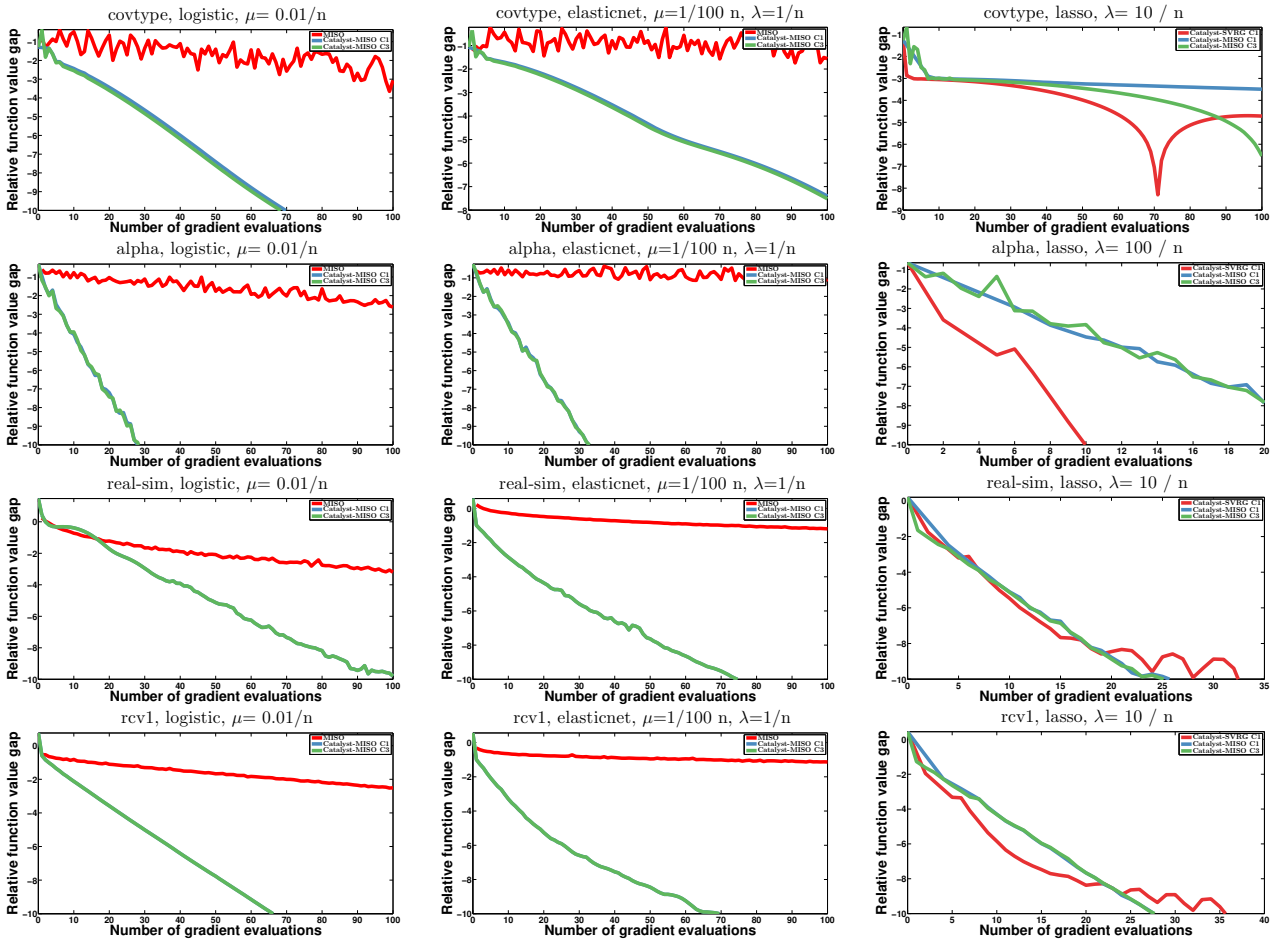

Figure 3.3 – Experimental study of different stopping criterions for Catalyst-MISO, with a similar setting as in Figure 3.1

3.6.4 Acceleration of Existing Methods

Then, we put the previous curves into perspective and make a comparison of the performance before and after applying Catalyst across methods. We show the best performance among the three developed stopping criteria, which corresponds to be (C3) .

Observations. We observe that by applying Catalyst, we accelerate the original algorithms up to the limitations discussed above (comparing the dashed line and the solid line of the same color). In three datasets (covtype, real-sim and rcv1), significant improvements are achieved as expected by the theory for the ill-conditioned problems in logistic regression and Elastic-net. For dataset alpha, we remark that an relative accuracy in the order 10^{-10} is attained in less than 10 iterations. This suggests that the problems is in fact well-conditioned and there is

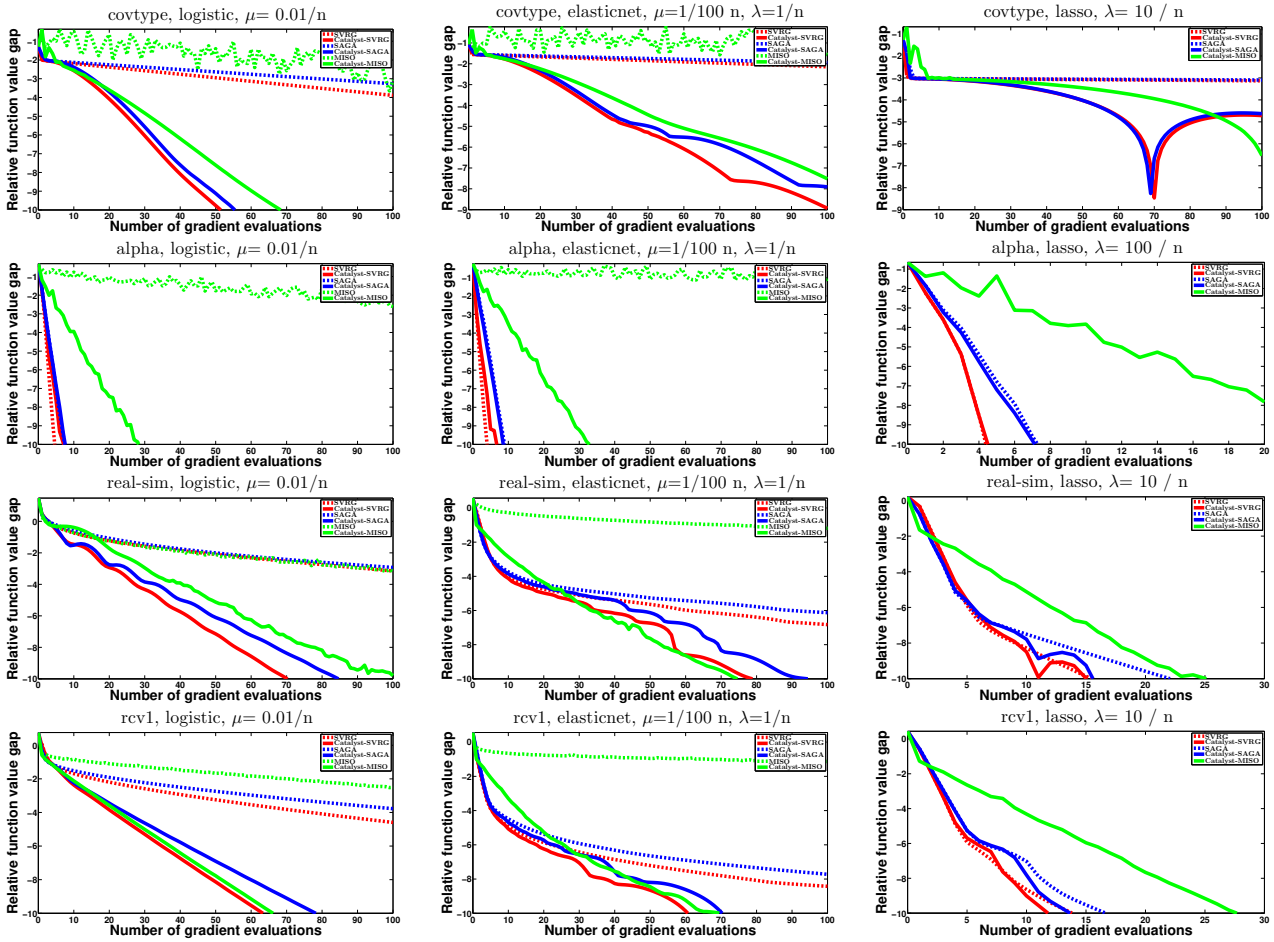

Figure 3.4 – Experimental study of the performance of Catalyst applying to SVRG, SAGA and MISO. The dashed lines correspond to the original algorithms and the solid lines correspond to accelerated algorithms by applying Catalyst. We plot the relative function value gap $(f(x_k) - f^*)/f^*$ in the number of gradient evaluations, on a logarithmic scale.

some hidden strong convexity for this dataset. Thus, the incremental algorithms like SVRG or SAGA are already optimal under this situation and no improvement is obtained by applying Catalyst.

3.6.5 Study of the Parameter κ

Finally, we provide evaluations of the performance for different κ.

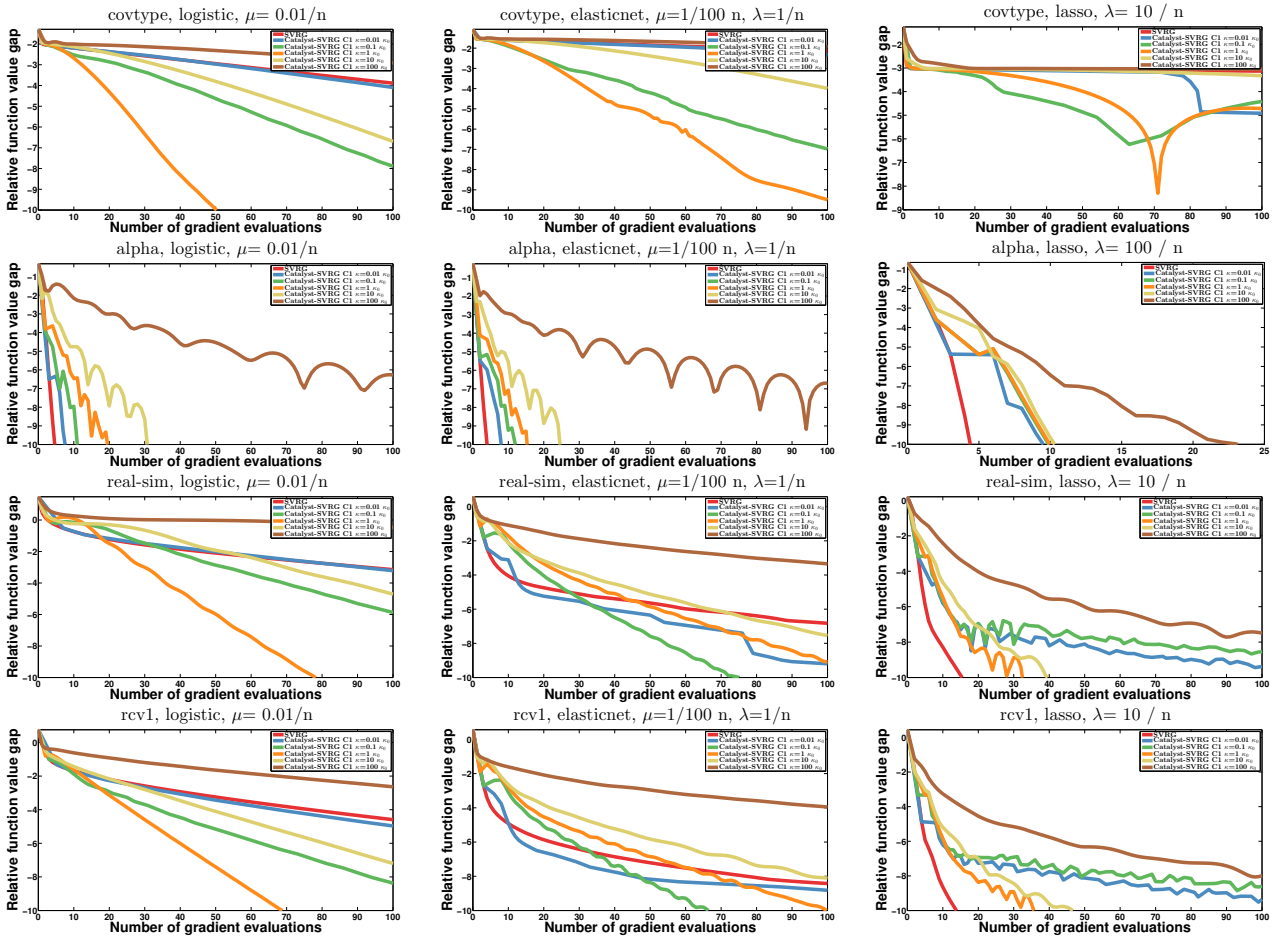

Figure 3.5 – Evaluations of Catalyst-SVRG for different κ using stopping criterion C1, where κ_0 is the theoretical choice given by the complexity analysis.

Observations for different choices of κ. We consider a logarithmic grid $\kappa = 10^i \kappa_0$ with $i = -2, -1, \cdots, 2$ and κ_0 is the optimal κ given by the theory. We observe that for ill-conditioned problems, using optimal choice κ_0 provides much better performance than other choices, which confirms the theoretical result. For the dataset of alpha or Lasso problems, we observe that the

best choice is given by the smallest $\kappa = 0.01\kappa_0$. This suggests that, as discussed before, strong convexity may be hidden in the loss for this dataset.

3.7 Discussions and concluding remarks

In this chapter, we have introduced a generic algorithm, Catalyst, that brings Nesterov's acceleration to a large class of first-order methods. In particular, it can be applied to previously unaccelerated incremental algorithms such as SAG/SAGA, SVRG, MISO/Finito. The resulting algorithms after applying Catalyst achieve the optimal complexity up to a logarithmic factor. We studied experimentally different variants of it and show the acceleration does occur in practice. We compare different choices of parameters and show that the theoretical choice provides promising performance, especially for ill-conditioned problems. The current version of the algorithm requires the knowledge of both the Lipschitz constant parameter and the strong convexity parameter. As we have seen in the experiments, the acceleration degrades when the parameter are not correctly estimating, such as the existence of hidden strong convexity. Designing a parameter free algorithm is one of our important future directions.

Chapter 4

QuickeNing: acceleration with Quasi-Newton principles

Chapter abstract:
We propose a generic approach to accelerate gradient-based optimization algorithms with quasi-Newton principles. The proposed scheme, called QuickeNing, can be applied the same class of methods as Catalyst, including classical gradient methods and incremental first-order methods. It is a variable metric approach compatible with composite objectives, meaning that it has the ability to deal with sparsity-inducing regularizations and provide exactly sparse solutions. QuickeNing relies on applying an inexact limited-memory BFGS rules to the Moreau-Yosida regularization, which is also known as variable metric proximal point algorithm. We provide a global convergence analysis, showing that QuickeNing enjoys a worst-case linear convergence rate for strongly convex problems. We present experimental results where QuickeNing gives significant improvements over competing methods for solving large-scale high-dimensional machine learning problems.

The material of this chapter is based on the following paper, which is currently under review:

H. Lin, J. Mairal, and Z. Harchaoui. A generic quasi-newton algorithm for faster gradient-based optimization. *arXiv:1610.00960v2*, 2017

The code for reproducing the figures in this chapter is publicly available at: https://github.com/hongzhoulin89/Catalyst-QNing

4.1 Introduction

First-order methods are often the default choice in machine learning for solving the empirical risk minimization problem,

$$\min_{x\in\mathbb{R}^p} \left\{ f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \psi(x) \right\}. \tag{4.1}$$

However, 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., [Schmidt et al., 2017] 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 [Liu and Nocedal, 1989, Nocedal, 1980]. The theoretical guarantees offered by L-BFGS are somewhat weak, meaning that it does not outperform accelerated first-order methods in terms of worst-case convergence rate, and also it is not guaranteed to correctly approximate the Hessian of the objective. Yet, it remains one of the greatest practical success of smooth optimization. Adapting L-BFGS to composite and structured problems, such as the finite sum of functions (4.1), is of utmost importance nowadays.

For instance, there have been several attempts to develop a proximal Quasi-Newton method [Byrd et al., 2015, Lee et al., 2012, Scheinberg and Tang, 2016, Yu et al., 2008]. These algorithms typically require computing many times the proximal operator of ψ with respect to a variable metric, which may be as computationally demanding as solving the original problem. Quasi-Newton steps were also incorporated as local search steps into accelerated first-order methods to further enhance their numerical performance [Ghadimi et al., 2015]. More related to our work, L-BFGS is combined with SVRG for minimizing smooth finite sums in [Gower et al., 2016]. 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 [Defazio et al., 2014a,b, Mairal, 2015, Schmidt et al., 2017, Shalev-Shwartz and Zhang, 2016] and block coordinate descent methods [Razaviyayn et al., 2013, Richtárik and Takáč, 2014]

More precisely, our main contribution is a generic meta-algorithm, called QuickeNing (the letters "Q" and "N" stand for Quasi-Newton), which uses a given optimization method to solve a sequence of auxiliary problems up to some appropriate accuracy, resulting in faster global convergence in practice. The resulting scheme admits a simple interpretation: *it may be seen as applying the L-BFGS algorithm with inexact (but accurate enough) gradients to the Moreau-Yosida regularization of the objective.* As a result, our approach is (i) generic, as stated previously; (ii) despite the smoothing of the objective, the sub-problems that we solve are composite ones, which may lead to exactly sparse iterates when a sparsity-inducing regularization is involved, e.g., the ℓ_1-norm; (iii) it admits a worst-case linear convergence rate for strongly convex problems similar to that of gradient descent, which is typically the best guarantees obtained for L-BFGS schemes in the literature; (iv) it is easy to use and does

not require using a line search algorithm, which is sometimes computationally expensive and difficult to calibrate in classical Quasi-Newton methods.

The idea of combining second-order or quasi-Newton methods with Moreau-Yosida regularization is in fact relatively old. It may be traced back to variable metric proximal bundle methods [Chen and Fukushima, 1999, Fukushima and Qi, 1996, Mifflin, 1996], which use BFGS updates on the Moreau-Yosida smoothing of the objective and bundle methods to approximately solve the corresponding sub-problems. Our approach revisits this principle with *a limited-memory variant* (to deal with large dimension d), with *an alternative strategy to line search schemes*—which is useful when f is a large sum of n functions as in (4.1)—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 QuickeNing on regularized logistic regression and regularized least-squares, with smooth and non smooth regularization penalities such as the Elastic-Net [Zou and Hastie, 2005]. We use large-scale machine learning datasets and show that QuickeNing performs at least as well as the recently proposed Catalyst [Lin et al., 2015] and as the classical L-BFGS scheme in all experiments, and significantly outperforms them in many cases.

This chapter is organized as follows: Section 4.2 presents related work on Quasi-Newton methods such as L-BFGS; we introduce QuickeNing in Section 4.3, and we provide a convergence analysis in Section 4.4; Section 4.5 is devoted to experiments and Section 4.6 concludes the chapter.

4.2 Related work and preliminaries

The history of quasi-Newton methods can be traced back to the 1950's [Bonnans et al., 2006, Hiriart-Urruty and Lemaréchal, 1996, Nocedal and Wright, 2006]. Quasi-Newton methods often lead to significantly faster convergence in practice compared to simpler gradient-based methods for solving smooth optimization problems [Schmidt et al., 2011a]. 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 recall the well-known BFGS algorithm in Section 4.2.1, its limited memory variant [Nocedal, 1980], and a few recent extensions. Then, we present earlier works that combine proximal point and Quasi-Newton algorithms in Section 4.2.2, which is related to the strategy we use in this chapter.

4.2.1 Quasi-Newton methods for smooth optimization

In Section 1.3, we introduce Quasi-Newton methods for smooth optimization. Here, we recall the updates of BFGS, named after its inventors (Broyden-Fletcher-Goldfarb-Shanno), and its limited variant L-BFGS [Nocedal and Wright, 2006]. These approaches will be the workhorses of the QuickeNing meta-algorithm. 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:

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k) \quad \text{and} \quad B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}, \tag{4.2}$$

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 [Nocedal and Wright, 2006].

The limited memory variant L-BFGS [Nocedal, 1980] 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 (4.2) 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, (see Section 1.3).

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 (4.1). This motivated stochastic counterparts of the Quasi-Newton method (SQN) [Byrd et al., 2016]. The direct application substituting the full gradient $\nabla f(x_k)$ by a stochastic estimate is unfortunately not convergent. Instead, the SQN method [Byrd et al., 2016] 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 [Gower et al., 2016, Moritz et al., 2016].

4.2.2 Combining the proximal point algorithm and Quasi-Newton

There are several ways to extend Quasi-Newton methods for nonsmooth optimization. A first approach consists in minimizing successive quadratic approximations, providing the update

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^p} \left\{ f_0(x_k) + \langle \nabla f_0(x_k), x - x_k \rangle + \frac{1}{2}(x - x_k)^T B_k (x - x_k) + \psi(x) \right\}, \tag{4.3}$$

which is known as proximal Quasi-Newton method [Byrd et al., 2015, Lee et al., 2012, Scheinberg and Tang, 2016, Yu et al., 2008]. A major drawback is then the computation of the proximal operator. Since B_k is dense and changes over the iterations, an exact solution to the proximal

operator is usually not available. An inexact variant has been analyzed in [Byrd et al., 2015], but it is unclear whether or not the quadratic approximation could be solved inexactly in an efficient manner.

A second approach consists in applying Quasi-Newton methods after Moreau-Yosida smoothing or equivalently a proximal point algorithm on the objective function [Bonnans et al., 1995, Fukushima and Qi, 1996, Chen and Fukushima, 1999, Burke and Qian, 2000, Fuentes et al., 2012]. The underlying idea is to first smooth and improve the conditioning of the objective function, then apply a Quasi-Newton method on the smoothed function. We recall here the definition of the Moreau-Yosida regularization, which we have presented in Section 1.6

$$F(x) \triangleq \min_{z \in \mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2} \|z - x\|^2 \right\}, \tag{4.4}$$

where κ is a positive scalar. Applying Quasi-Newton methods on F gives a sequence of subproblems in the shape of (4.4). The main difference between the proximal Newton approach and this smoothing approach is that the subproblems generated by (4.3) change the metric at every iteration while the subproblems in (4.4) have a fixed conditioning. Moreover, the conditioning of the subproblems is explicitly controlled by the parameter κ, which is free to choose. Thus, it suffices to take an appropriate value of κ to make the subproblem well conditioned and apply an optimization method to approximate the solution. For instance, in [Chen and Fukushima, 1999], the sub-problems are solved inexactly by proximal bundle methods, and in [Fuentes et al., 2012] by using a sophisticated sufficient descent criterion. While the proposed algorithms are elegant, the required complexity of solving these subproblems has not been investigated, which explains the lack of global complexity result.

The proposed QuickeNing algorithm, which will be presented in the next section, follows this line of research. We provide a global convergence analysis taking into account the complexity for solving the subproblems. More importantly, the complexity result allows us to provide an explicit choice for the parameter κ, which hardly discussed in the previous work. Moreover, we propose a simple strategy to set the parameters, removing the need of a line-search scheme in the determination of stepsizes. Several theoretical grounded warm start strategies are proposed for efficiently solving the sub-problems, leading to a significant acceleration in practice.

4.3 QuickeNing: a Quasi-Newton meta-algorithm

We now present the QuickeNing method in Algorithm 5, which consists of applying L-BFGS rules 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 L-BFGS 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.

Algorithm 5 QuickeNing

input Initial point x_0 in $\mathbb{R}^p$; number of iterations K; parameter $\kappa > 0$; minimization algorithm $\mathcal{M}$.

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

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

4: Estimate the new gradient and the Moreau-Yosida function value

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

5: **if** sufficient approximate decrease is obtained

$$F_{\text{test}} \leq F_k - \frac{1}{2\kappa}\|g_k\|^2, \tag{4.5}$$

then
6: Accept the new iterate: $(x_{k+1}, g_{k+1}, F_{k+1}, z_{k+1}) = (x_{\text{test}}, g_{\text{test}}, F_{\text{test}}, z_{\text{test}})$.
7: **else**
8: Update the current iterate with the proximal mapping: $x_{k+1} = z_k$.

$$(g_{k+1}, F_{k+1}, z_{k+1}) = \mathsf{ApproxGradient}\,(x_{k+1}, \mathcal{M})\ .$$

9: **end if**
10: update $H_{k+1} = \mathsf{L\text{-}BFGS}(H_k, x_{k+1} - x_k, g_{k+1} - g_k)$.
11: **end for**

output last proximal mapping z_K (solution).

Algorithm 6 Generic procedure ApproxGradient

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

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

$$z \approx \arg\min_{w \in \mathbb{R}^p} \left\{ h(w) \triangleq f(w) + \frac{\kappa}{2}\|w - x\|^2 \right\}, \tag{4.6}$$

using the warm start strategies and stopping criterions described later in this section.
2: 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.

4.3.1 The main algorithm

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

Outer-loop: inexact L-BFGS. The L-BFGS rule 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 Algorithm 7.4 of [Nocedal and Wright, 2006] 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 [Friedlander and Schmidt, 2012]).

Inner-loop: approximate Moreau-Yosida envelope. The minimization algorithm $\mathcal{M}$ is used to compute an approximate Moreau-Yosida envelope of the objective. The procedure ApproxGradient () calls the minimization algorithm $\mathcal{M}$ to minimize the sub-problem (4.6). When the problem is solved exactly, the function returns the exact values $g = \nabla F(x)$, $F_a = F(x)$ and $z = p(x)$. Otherwise, a stopping criterion should be used. In this chapter, we consider the three following cases:

(a) defining a pre-defined decreasing sequence $(\varepsilon_k)_{k\geq 0}$ and stopping criterion based on function values—that is, stop the optimization method $\mathcal{M}$ when the approximate solution of (4.6) satisfies $h(z) - h^* \leq \varepsilon_k$, where $h^* = \min_{z\in\mathbb{R}^d} h(z)$, which may be checked by computing a duality gap. This is the less practical of the three strategies presented here.

(b) defining an adaptive sequence $(\varepsilon_k)_{k\geq 0}$, which depends on quantities that are available at iteration k.

(c) using a pre-defined budget $T_{\mathcal{M}}$ in terms of number of iterations of the method $\mathcal{M}$ at iteration k.

Section 4.4 will provide theoretical and practical insight about these parameter choices. Before that, we discuss the requirements on f and $\mathcal{M}$ and warm start strategies to solve the sub-problems.

Requirements on f. When f is μ-strongly-convex, we show that the method achieves a linear convergence rate (which takes into account the complexity of solving the sub-problems with $\mathcal{M}$); to obtain this guarantee, the accuracy for solving the sub-problems needs to decrease at the same linear rate. In principle, all strategies (a)-(b)-(c) above may be used to stop $\mathcal{M}$, as discussed in Section 4.4.

When f is convex but not strongly convex (meaning $\mu = 0$), it is possible to achieve the classical $O(1/k)$ worst-case convergence rate on function values, by using a particular strategy (b) and by replacing the approximate descent condition (4.5) by a slightly stronger one $f(x_{\text{test}}) \leq f(z_k)$, see Proposition 15 for more details.

Requirements on $\mathcal{M}$ and warm start. We have mentioned that QuickeNing applies to methods $\mathcal{M}$ with linear convergence rates for strongly-convex problems. More precisely, we consider methods $\mathcal{M}$ that are always able to produce a sequence of iterates $(w_t)_{t\geq 0}$ for solving each sub-problem (4.6) such that

$$h(w_t) - h^* \leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^t(h(w_0) - h^*) \quad \text{for some constants} \quad C_{\mathcal{M}}, \tau_{\mathcal{M}} > 0, \tag{4.7}$$

where w_0 is the initial point given to $\mathcal{M}$. When (4.7) is satisfied, we call $\mathcal{M}$ a *primal* method. QuickeNing can also afford non-deterministic methods $\mathcal{M}$ that satisfy (4.7) in expectation; in such a case, our complexity results also hold in expectation. The condition typically holds for many primal gradient-based optimization techniques, such as block-coordinate descent algorithms [Nesterov, 2012b, Richtárik and Takáč, 2014], or some incremental algorithms [Defazio et al., 2014a, Schmidt et al., 2017]. The warm start strategy derived from our convergence analysis in Section 4.4 is simply $w_0 = x_{\text{test}}$. The class of *dual* methods MISO/Finito [Defazio et al., 2014b, Mairal, 2015] can also be applied with a different warm start in the same way as Catalyst [Lin et al., 2015]. Finally, before giving more precise theoretical statements, we discuss briefly two important features of the algorithm.

Handling composite objective functions. In machine learning or signal processing, convex composite objectives (4.1) 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 [Nesterov, 2005] may allow us to solve the optimization problem up to some chosen accuracy, but they provide solutions that do not inherit the properties induced by the non-smoothness of the objective. To illustrate what we mean by this sentence, we may consider smoothing the ℓ_1-norm, leading to a solution vector with small coefficients, but not with exact zeroes. When the goal is to perform model selection—that is, understanding which variables are important to explain a phenomenon, exact sparsity is seen as an asset, and optimization techniques dedicated to composite problems such as FISTA [Beck and Teboulle, 2009] are often preferred.

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 4.4 studies the convergence of the sequence $(f(z_k))_{k\geq 0}$ to the solution f^*.

On the absence of line-search scheme. Another key property of the QuickeNing algorithm is that it does not require using a line-search scheme to select a step-size, which is typically necessary to ensure the monotonic descent (and convergence) of classical BFGS and L-BFGS algorithms [Nocedal, 1980]. In the context of the Moreau-Yosida regularization, any line-search

would be prohibitive, since it would require to evaluate the function F multiple times, hence solving the sub-problems (4.6) as many times at every iteration.[1] Here, we choose a simple strategy that selects $x_{k+1} = z_k$ when the sufficient descent condition (4.5) is not satisfied. z_k is indeed a good candidate since it is obtained by performing one step of the inexact proximal point algorithm, which we have described above, and whose convergence properties are well understood. In practice, we have observed that the sufficient decrease condition is almost all the time satisfied, given the choice of parameters provided in the experimental section.

4.4 Convergence and complexity analysis

In this section, we study the convergence of the QuickeNing algorithm—that is, the rate of convergence to zero of the quantities $(f(z_k) - f^*)_{k\geq 0}$ and $(F(x_k) - F^*)_{k\geq 0}$, and also its computational complexity, which takes into account the cost of solving the sub-problems (4.6). We consider several cases, leading to different variants of the parameter choices, warm start and approximation strategies. We start by stating the main properties of the gradient approximation in Section 4.4.1. Then, we analyze the convergence of the outer loop algorithm in Section 4.4.2, and then Section 4.4.3 is devoted to the global complexity analysis.

4.4.1 Properties of the gradient approximation

The next lemma is classical and provides approximation guarantees about the quantities returned by the ApproxGradient procedure (Algorithm 6); see [Bertsekas, 2015, Fukushima and Qi, 1996]. We recall here the proof for completeness.

Lemma 11 (Approximation quality of the gradient approximation). *Consider a vector x in $\mathbb{R}^p$, a positive scalar ε and compute*

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

such that $h(z) - h^* \leq \varepsilon$, where $h^* = \min_{v\in\mathbb{R}^p} h(v)$. As in Algorithm 6, define $g = \kappa(x - z)$ and $F_a = h(z)$. Then, the following inequalities hold

$$F(x) \leq F_{\mathrm{a}} \leq F(x) + \varepsilon, \tag{4.8}$$

$$\|z - p(x)\| \leq \sqrt{\frac{2\varepsilon}{\kappa}}, \tag{4.9}$$

$$\|g - \nabla F(x)\| \leq \sqrt{2\kappa\varepsilon}. \tag{4.10}$$

[1] Note that in the context of BFGS applied to F, a heuristic line-search that requires evaluating f instead of F is also proposed in [Chen and Fukushima, 1999], but this heuristic does not guarantee the algorithm convergence and is still computationally demanding when f is a large sum of functions as in (4.1).

Proof. (4.8) is straightforward by definition of $h(z)$. Since f is convex, the function h is κ-strongly convex, and (4.9) 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 (4.10) 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 (1.37). □

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

Lemma 12 (Simple inequalities regarding the gradient approximation). *Consider the same quantities introduced in Lemma 11. Then,*

$$f(z) = F_{\mathrm{a}} - \frac{1}{2\kappa}\|g\|^2, \tag{4.11}$$

$$\frac{1}{2}\|\nabla F(x)\|^2 - 2\kappa\varepsilon \leq \|g\|^2 \leq 2(\|\nabla F(x)\|^2 + 2\kappa\varepsilon). \tag{4.12}$$

Proof. Eq. (4.11) is obtained by using the definitions of g and noticing that $F_{\mathrm{a}} = h(z)$. Eq. (4.12) follows from

$$\begin{aligned} \|\nabla F(x)\|^2 &\leq 2(\|\nabla F(x) - g\|^2 + \|g\|^2) \\ &\leq 2(2\kappa\varepsilon + \|g\|^2) \qquad \text{(from (4.10))}. \end{aligned}$$

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

4.4.2 Convergence analysis of the outer loop

We are now in shape to establish the convergence of the QuickeNing meta-algorithm, without considering yet the cost of solving the sub-problems (4.6). We first remark that the following relation always holds

$$(g_k, F_k, z_k) = \mathsf{ApproxGradient}(x_k, \mathcal{M}) \;, \tag{4.13}$$

which allows us to apply Lemma 12 for all $k \geq 0$. In the following analysis, we will always call ε_k the precision (in the sense of Lemma 11) of solving the sub-problem (4.6) related to the call (4.13).

Lemma 13 (Approximate descent property). *Consider the sequences $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ generated by Algorithm 5 where z_k is obtained by solving a sub-problem (4.6) with accuracy ε_k. Then,*

$$F(x_{k+1}) \leq f(z_k) \leq F(x_k) - \frac{1}{4\kappa}\|\nabla F(x_k)\|^2 + 2\varepsilon_k. \tag{4.14}$$

Proof. By (4.11), we have

$$f(z_k) = F_k - \frac{1}{2\kappa}\|g_k\|^2.$$

To show the first inequality, we consider two cases:

- When the condition (4.5) is violated, then we set $x_{k+1} = z_k$. As a result,

$$F(x_{k+1}) = \min_{z\in\mathbb{R}^k}\left\{f(z) + \frac{\kappa}{2}\|z - x_{k+1}\|^2\right\} \leq f(x_{k+1}) = f(z_k).$$

- Otherwise, we have $F(x_{k+1}) \leq F_{k+1} \leq F_k - \frac{1}{2\kappa}\|g_k\|^2 = f(z_k)$.

Thus, we have $F(x_{k+1}) \leq f(z_k)$ in both cases and we may now prove the second inequality:

$$\begin{aligned} f(z_k) &= F_k - \frac{1}{2\kappa}\|g_k\|^2 \\ &\leq F(x_k) + \varepsilon_k - \left(\frac{1}{4\kappa}\|\nabla F(x_k)\|^2 - \varepsilon_k\right) \quad \text{(from (4.8) and (4.12))} \\ &= F(x_k) - \frac{1}{4\kappa}\|\nabla F(x_k)\|^2 + 2\varepsilon_k. \end{aligned}$$

This concludes the proof. □

This lemma gives us a first intuition about the natural choice of the accuracy ε_k to use, which should be of the same order as $\|\nabla F(x_k)\|^2$. Unfortunately, the quantity $\|\nabla F(x_k)\|^2$ is not known since computing its value requires solving exactly the subproblem (4.6). We propose in the next section several practical strategies to set the sequences $(\varepsilon_k)_{k\geq 0}$. Before that, we now use the approximate descent property of the previous lemma to control the accumulation of errors across iterations.

Strongly-convex objectives.

We start by analyzing the convergence of our algorithm for strongly-convex objectives.

Proposition 12 (Convergence of Algorithm 5 with theoretical sequence $(\varepsilon_k)_{k\geq 0}$.). *Assume that f is μ-strongly convex and define $\rho = \frac{\mu}{4(\mu+\kappa)}$. Consider the sequences $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ generated by Algorithm 5 where z_k is obtained by solving a sub-problem (4.6) with accuracy ε_k where*

$$\varepsilon_k \leq \frac{1}{16\kappa}\|\nabla F(x_k)\|^2. \tag{4.15}$$

Then,

$$F(x_{k+1}) - F^* \leq f(z_k) - f^* \leq (1-\rho)^{k+1}\left(f(x_0) - f^*\right).$$

Proof. We may apply Lemma 13, and from (4.14), we obtain

$$\begin{aligned} F(x_{k+1}) - F^* \leq f(z_k) - F^* &\leq F(x_k) - F^* - \frac{1}{8\kappa}\|\nabla F(x_k)\|^2 \\ &\leq F(x_k) - F^* - \frac{\mu_F}{4\kappa}(F(x_k) - F^*) \\ &= (1-\rho)\,(F(x_k) - F^*) \\ &\leq (1-\rho)^{k+1}\,(f(x_0) - f^*), \end{aligned}$$

where the second inequality uses a classical relation Theorem 2.1.10 in [Nesterov, 2004] for strongly-convex functions, and the third one simply uses the definition of ρ given in Proposition 14. □

The previous proposition is interesting from a theoretical point of view, but, as discussed previously, the quantity $\|\nabla F(x_k)\|^2$ is unknown in advance. Therefore, the above criterion does not seem practical at first sight. The next proposition presents one way to use them concretely.

Proposition 13 (On using Proposition 12 in practice). *The following condition implies the inequality (4.15) in Proposition 12:*

$$\varepsilon_k \leq \frac{1}{36\kappa}\|g_k\|^2.$$

Proof. We may use (4.12) from Lemma 12, which gives

$$\|g_k\|^2 \leq 2\left(\|\nabla F(x_k)\|^2 + \frac{1}{18}\|g_k\|^2\right),$$

which implies

$$\frac{4}{9}\|g_k\|^2 \leq \|\nabla F(x_k)\|^2 \quad \text{and thus} \quad \varepsilon_k \leq \frac{1}{36\kappa}\|g_k\|^2 \leq \frac{1}{16\kappa}\|\nabla F(x_k)\|^2.$$

□

While we do not have access to $\|\nabla F(x_k)\|^2$, we do have access to the estimate gradient $g_k = \kappa(x_k - z_k)$. This immediately suggests the following stopping criterion when applying the method $\mathcal{M}$ to (4.6)

$$h(w_t) - h^* \leq \frac{\kappa}{36}\|w_t - x\|^2 \tag{4.16}$$

where $(w_t)_{t\geq 0}$ is the sequence produced by $\mathcal{M}$ to solve the sub-problem (4.6). Again, such a condition can often be checked by computing duality gap. This choice ensures the linear convergence of the sequences $(F(x_k))_{k\geq 0}$ and $(f(z_k))_{k\geq 0}$ to f^*. We shall also see in Section 4.4.3 that the condition may also be enforced by simply using a constant number of iterations of the method $\mathcal{M}$ when the right warm start strategy is used. Another possibility is to use a predefined sequence $(\varepsilon_k)_{k\geq 0}$, similar to Catalyst [Lin et al., 2015].

Proposition 14 (Convergence of Algorithm 5 with pre-defined sequence). *Assume that f is μ-strongly convex and define $\rho = \frac{\mu}{4(\mu+\kappa)}$. Consider the sequences $(x_k)_{k\geq 0}$ and $(z_k)_{k\geq 0}$ generated by Algorithm 5 where z_k is obtained by solving a sub-problem (4.6) with accuracy ε_k for all k. Then,*

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

Proof. First, we recall from Proposition 1 that F is μ_F-strongly convex with parameter $\mu_F = \frac{\mu\kappa}{\mu+\kappa}$ and that the optimal value of the objectives f and F coincide—in other words, and $F^* = f^*$. We may then apply Lemma 13 and

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

where the last inequality simply comes from the upper-bound $F \leq f$. □

Corollary 5 (Recommended pre-defined sequence $(\varepsilon_k)_{k\geq 0}$). *If the sequence $(\varepsilon_k)_{k\geq 0}$ in Proposition 14 satisfies*

$$\varepsilon_k = \frac{1}{2}C(1-\rho)^{k+1} \qquad \text{with} \qquad C \geq f(x_0) - f^*.$$

then

$$F(x_{k+1}) - F^* \leq f(z_k) - f^* \leq \frac{C}{\rho}\,(1-\rho)^{k+2}\,. \tag{4.17}$$

Proof. From Proposition 14, we have

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

□

We remark that, of course, the quantity $f(x_0) - f^*$ is unknown in practice, but the proposition only requires an upper-bound, e.g., a duality gap, or simply $f(x_0)$ if the function is non-negative. Next, we present a similar complexity analysis for convex, but not strongly-convex objectives.

Convex but not strongly-convex objectives.

The goal of this section is to show that a minor modification of the algorithm yields the classical $O(1/k)$ convergence rate of the sequences $(F(x_k))_{k\geq 0}$ and $(f(z_k))_{k\geq 0}$, which can be shown to be slightly stronger than the previous one, when the accuracy ε_k is small enough. Specifically, we replace the descent condition (4.5) by $f(x_{\text{test}}) \leq f(z_k)$. We may now use the new descent condition to derive a convergence rate for non-strongly convex function.

Proposition 15 (Convergence of Algorithm 5 for convex, but not strongly-convex objectives). *Let f be a convex function, which attains its minimum at x^*. Assume that the level sets of f are bounded. More precisely, assume that there exists $R > 0$ such that*

$$\forall y \in \mathbb{R}^p \quad s.t. \quad f(y) \leq f(x_0) \quad then \quad \|y - x^*\| \leq R.$$

Assume that the sub-problems (4.6) are solved up to accuracy $\varepsilon_k \leq \frac{1}{2\kappa}\|g_k\|^2$ for any k, and replace the descent condition (4.5) in Algorithm 5 by $f(x_{test}) \leq f(z_k)$; then, the sequence $(x_k)_{k\geq 1}$ satisfies

$$f(x_k) - f^* \leq \frac{2\kappa R^2}{k+3}.$$

Proof. First, we remark that with the new condition, we always have

$$f(x_{k+1}) \leq f(z_k),$$

since $x_{k+1} = z_k$ when the descent condition $f(x_{\text{test}}) \leq f(z_k)$ is not satisfied. Moreover, according to Lemma 11,

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

and then, by combining the two previous inequalities

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

This relation allows us to use a similar proof technique introduced in [Nesterov, 2013] in the context of the proximal gradient descent method

$$\begin{aligned} f(x_{k+1}) \leq F(x_k) &= \min_{z\in\mathbb{R}^p} \left\{ f(z) + \frac{\kappa}{2}\|z - x_k\|^2 \right\} \\ &\leq \min_{\alpha\in[0,1]} \left\{ f(\alpha x^* + (1-\alpha)x_k) + \frac{\kappa\alpha^2}{2}\|x^* - x_k\|^2 \right\} \\ &\leq \min_{\alpha\in[0,1]} \left\{ \alpha f(x^*) + (1-\alpha) f(x_k) + \frac{\kappa\alpha^2}{2}\|x^* - x_k\|^2 \right\}. \end{aligned}$$

Since $f(x_k)$ is necessarily decreasing (take $\alpha = 0$ in the previous relation), we may use the bounded level set assumption, which leads to

$$f(x_{k+1}) - f^* \leq \min_{\alpha\in[0,1]} \left\{ (1-\alpha)(f(x_k) - f^*) + \frac{\kappa\alpha^2 R^2}{2} \right\}.$$

1. If $f(x_k) - f^* \geq \kappa R^2$, then $\alpha^* = 1$ and $f(x_{k+1}) - f^* \leq \frac{\kappa R^2}{2}$.
2. Otherwise, $\alpha^* = \frac{f(x_k)-f^*}{\kappa R^2}$, which gives

$$r_{k+1} \leq r_k(1 - \frac{r_k}{2\kappa R^2}) \leq \frac{\kappa R^2}{2},$$

where $r_k \triangleq f(x_k) - f^*$. Thus

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

In both cases, $r_1 \leq \frac{\kappa R^2}{2}$ and there

$$\frac{1}{r_{k+1}} \geq \frac{1}{r_1} + \frac{k}{2\kappa R^2} \geq \frac{k+4}{2\kappa R^2},$$

which is sufficient to conclude. □

4.4.3 Complexity analysis of the inner loop

In this section, we compute the complexity of solving the sub-problems (4.6) using the method $\mathcal{M}$, which allows us to obtain the global complexity of QuickeNing, when using the various approximation strategies described in the previous section. We analyze here primal approaches $\mathcal{M}$, which achieve the convergence guarantees (4.7). Dual methods like MISO/Finito whose convergence rate does not satisfy (4.7) require a different warm start as presented in the appendix of [Lin et al., 2017]. In both cases, our main result is that all sub-problems can

be solved with an appropriate accuracy in a constant number $T_{\mathcal{M}}$ of iterations. One of the key is to choose the right warm start point, as detailed next. The first lemma characterizes the quality of the initialization $w_0 = x$ for solving the sub-problem (4.6) when the objective function is smooth.

Lemma 14 (**Warm start for primal methods - smooth case**). *If f is differentiable with L-Lipschitz continuous gradients, we can initialize the method $\mathcal{M}$ with $w_0 = x$ for solving (4.6). Then, we have the guarantee that*

$$h(w_0) - h^* \leq \frac{L+\kappa}{2\kappa^2}\|\nabla F(x)\|^2. \tag{4.18}$$

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 14 relies on the smoothness of f, which does not hold for composite problems. The next lemma addresses this issue.

Lemma 15 (**Warm start for primal methods - composite case**). *Consider the composite optimization problem (4.1). We may initialize the method $\mathcal{M}$ for solving (4.6) with*

$$w_0 = \arg\min_{w \in \mathbb{R}^p} \left\{ f_0(x) + \langle \nabla f_0(x), w - x\rangle + \frac{L+\kappa}{2}\|w - x\|^2 + \psi(w) \right\}. \tag{4.19}$$

Then,

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

Proof. We use the inequality corresponding to Lemma 2.3 in [Beck and Teboulle, 2009]: for any w,

$$h(w) - h(w_0) \geq \frac{L'}{2}\|w_0 - x\|^2 + L'\langle w_0 - x, x - w\rangle, \tag{4.20}$$

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

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

□

The lemmas presented above are important, since they allow us to derive the global complexity of QuickeNing in the scenarios described in Propositions 12, 13, and 15. Specifically, it shows us that the conditions in both cases will be satisfied in a constant number of iterations of the method $\mathcal{M}$.

Proposition 16 (Inner-loop complexity for Algorithm 5). *Assume that f is differentiable with L-Lipschitz continuous gradients. Consider Algorithm 5 with the warm start strategy described in Lemma 14 or in Lemma 15. Assume that the primal method $\mathcal{M}$ applied in the inner loop produces a sequence $(w_t)_{t\geq 0}$ for each sub-problem (4.6) such that*

$$h(w_t) - h^* \leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^t(h(w_0)-h^*) \quad \text{for some constants} \quad C_{\mathcal{M}}, \tau_{\mathcal{M}} > 0. \tag{4.21}$$

Then, the sub-problems are solved with enough accuracy to apply Propositions 12, 13, and 15 in at most $T_{\mathcal{M}}$ iterations with

$$T_{\mathcal{M}} = \frac{1}{\tau_{\mathcal{M}}} \log\left(38 C_{\mathcal{M}} \frac{L+\kappa}{\kappa}\right).$$

Proof. The proposition is a direct consequence of the warm start strategy in previous lemmas. Consider at iteration k, we want to approximate the proximal mapping according to x_k. With the given $T_{\mathcal{M}}$ (which we abbreviate by T), we have

$$\begin{aligned} h(w_T) - h^* &\leq C_{\mathcal{M}}(1-\tau_{\mathcal{M}})^T(h(w_0)-h^*) \\ &\leq C_{\mathcal{M}} e^{-\tau_{\mathcal{M}} T}(h(w_0)-h^*) \\ &\leq C_{\mathcal{M}} e^{-\tau_{\mathcal{M}} T} \frac{L+\kappa}{2\kappa^2}\|\nabla F(x_k)\|^2 \quad \text{(By Lemma 14 and Lemma 15)} \\ &= \frac{1}{76\kappa}\|\nabla F(x_k)\|^2. \end{aligned}$$

It remains to show that this accuracy is in fact small enough to apply Propositions 12, 13, and 15. From Lemma 12, we know that $\|g_k\|$ and $\|\nabla F(x_k)\|$ are related through:

$$\frac{1}{2}\|\nabla F(x_k)\|^2 \leq \|g_k\|^2 + 2\kappa\varepsilon_k \leq \|g_k\|^2 + \frac{1}{38}\|\nabla F(x_k)\|^2.$$

Thus

$$h(w_T) - h^* \leq \frac{1}{76\kappa}\|\nabla F(x_k)\|^2 \leq \frac{1}{36\kappa}\|g_k\|^2.$$

This is exactly the accuracy required in Proposition 13. From which we easily derived the accuracy in Propositions 12 and 15.

□

4.4.4 Global complexity of QuickeNing

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

Proposition 17 (Worst-case global complexity for Algorithm 5). *Under the setting of Proposition 16, the number of iterations of the method $\mathcal{M}$ (or expected number if $\mathcal{M}$ is non-deterministic) to guarantee the optimality condition $f(z_k) - f^* \leq \varepsilon$ is*

- *for μ-strongly-convex problems:*

$$2T_{\mathcal{M}} \times \frac{\mu+\kappa}{\mu} \log\left(\frac{f(x_0)-f^*}{\varepsilon}\right) = 2\frac{\mu+\kappa}{\tau_{\mathcal{M}}\mu} \log\left(\frac{f(x_0)-f^*}{\varepsilon}\right) \log\left(38C_{\mathcal{M}}\frac{L+\kappa}{\kappa}\right).$$

- *for convex but not strongly convex problems:*

$$2T_{\mathcal{M}} \times \frac{2\kappa R^2}{\varepsilon} = \frac{4\kappa R^2}{\tau_{\mathcal{M}}\varepsilon} \log\left(38C_{\mathcal{M}}\frac{L+\kappa}{\kappa}\right).$$

Proof. Since in the worst case, there are two proximal problems to be solved at each outer-loop iteration, the total number of calls of method $\mathcal{M}$ is at most $2T_{\mathcal{M}}$ times the number of outer-loop iterations. The result follows immediately from Propositions 12 and 15. □

We give below the worst-case global complexity of QuickeNing when applied to two optimization methods $\mathcal{M}$ of interest. As we shall see, in terms of worst-case complexity bounds, QuickeNing does not lead to an improved convergence rate. 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. When comparing the vanilla L-BFGS algorithm with the gradient descent method, one outperforms the other in many practical cases, but never in theory.

Proposition 17 and its application to the two examples below show that, in terms of worse-case complexity, the QuickeNing scheme leaves the convergence rate unchanged. Linearly-converging methods remain linear-converging when used within the QuickeNing scheme.

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 $L/\mu \log(1/\varepsilon)$ iterations. Quickening-GD minimizes f to ε accuracy in $2(L+\kappa)/\mu \log(1/\varepsilon)$ iterations (when ignoring logarithmic terms).*

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. QuickeNing-SVRG with minimizes f to ε accuracy in

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

iterations in expectation.

Choice of κ. The worst-case complexity theory also does not seem to offer any guidance regarding the choice of κ for the QuickeNing acceleration to be most effective. We shall experiment with a heuristic, consisting in choosing the κ that would lead to the most acceleration for the related Catalyst acceleration scheme [Lin et al., 2015]. We present empirical evidence in support of this heuristic in Section 4.5.

4.5 Experiments and practical details

In this section, we present experimental results obtained by applying QuickeNing on SVRG (Section 4.5.3) and on the proximal gradient descent algorithm ISTA (Section 4.5.4), and we compare the resulting methods to other acceleration techniques and to the L-BFGS algorithm. We also study the behavior of QuickeNing when changing the smoothing parameter κ and the limited-memory parameter l (Section 4.5.5). Before that, we first present in Section 4.5.1 the formulations and datasets that we use, and also discuss in Section 4.5.2 various experimental and practical choices.

4.5.1 Formulations and datasets

We consider the same problems as in Section 3.6.1 including regularized logistic regression, Elastic-net and Lasso problem. The datasets are also identical to the that in the experiments of Catalyst with different characteristics in terms of size and dimension:

name	covtype	alpha	real-sim	rcv1
n	581 012	250 000	72 309	781 265
d	54	500	20 958	47 152

4.5.2 Choice of hyper-parameters and variants

Before presenting the numerical results, we discuss the choice of default parameters used in the experiments as well as different variants.

Choice of method $\mathcal{M}$. We consider the acceleration of the proximal SVRG algorithm [Xiao and Zhang, 2014] since it is able to adapt to the problem structure we consider: large sum of functions and composite regularization. We also consider accelerating the ISTA algorithm [Beck and Teboulle, 2009], which allow us to perform a comparison with the natural baselines FISTA and L-BFGS for smooth problems.

Stopping criteria for the inner loop. The default stopping criteria is to solve the sub-problem until accuracy ε_k such that $\varepsilon_k \leq \frac{1}{36\kappa}\|g_k\|^2$, which reduces to the simple criterion (4.16). Although we have shown that such accuracy is attainable in constant $T_{\mathcal{M}}$ iteration, the theoretical value of $T_{\mathcal{M}}$ could be very large depending on the conditioning of the problem. A natural heuristic is to always perform exactly one pass over the data in the inner loop without checking any stopping criteria as proposed in [Lin et al., 2015].

When applying QuickeNing to SVRG and ISTA, we call the one-pass variant **QuickeNing-SVRG1** and **QuickeNing-ISTA1**, respectively, and the one using stopping criterion (4.16) **QuickeNing-SVRG2** and **QuickeNing-ISTA2**.

	stopping criterion for $\min_w h(w)$
QuickeNing1	one-pass through the data.
QuickeNing2	stop when $h(w_t) - h^* \leq \frac{\kappa}{36}\|w_t - x_k\|^2$.

We upper bound the duality gap $h(w_t) - h^*$ either by evaluating the Fenchel conjugate or by computing the squared norm of the gradient. Finally, another variant that could be considered is to give a pre-defined sequence of $(\varepsilon_k)_{k\geq 0}$ and stop when the current accuracy is smaller than ε_k, which is checked by using Fenchel conjugate; typically, this variant is less practical and is thus omitted from the experiments.

Choice of regularization parameter κ. As the global complexity is a function of κ, the most intuitive choice is to choose κ to minimize the complexity. However, such analysis leads to $\kappa = 0$, which is not quite helpful since this suggests directly performing $\mathcal{M}$ on the original problem. The reason is that the convergence of L-BFGS is hard to characterize and theoretical rates of convergence can be pessimistic. Noting that for smooth functions, L-BFGS often outperforms Nesterov's accelerated gradient method, it is reasonable to expect QuickeNing achieves a similar complexity bound as Catalyst.

Choose κ to maximize $\dfrac{\tau_{\mathcal{M}}}{\sqrt{\mu + \kappa}}$,

Therefore, we choose κ identical as in Catalyst algorithm, which leads to $\kappa = L$ for gradient descent and $\kappa = L/2n$ for SVRG. Later in this section, a comparison of different scaling of κ is performed in order to demonstrate the relevance of this strategy.

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

Sufficient decrease condition. We use the sufficient decrease condition suggested by the theory:

- for strongly convex objectives, we use $F_{\text{test}} \leq F_k - \frac{1}{2\kappa}\|g_k\|^2$;
- for convex, but non-strongly convex objectives, we use $f(x_{k+1}) \leq f(z_k)$. Nevertheless, we will also evaluate experimentally the strategy for strongly-convex objectives in this context.

Comparison metric. For all experiments, we use the number of gradient evaluations as a natural metric, 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 [Nocedal and Wright, 2006], whereas evaluating the gradient of the full objective costs $O(nd)$, with $d \leq n$ and $l \ll d$. A similar choice was made in other works [Schmidt et al., 2017].

4.5.3 QuickeNing-SVRG for minimizing large sums of functions

We now apply QuickeNing to SVRG and compare different variants.

- **SVRG**: the Prox-SVRG algorithm of [Xiao and Zhang, 2014] 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 4.5.2.
- **Catalyst-SVRG**: The catalyst meta-algorithm of [Lin et al., 2015] applied to Prox-SVRG;
- **L-BFGS** (for smooth objectives): 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 [Schmidt et al., 2017]. The limited memory parameter l is also set to 100. We use it for the logistic regression experiment since L-BFGS is defined only for smooth objectives.

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

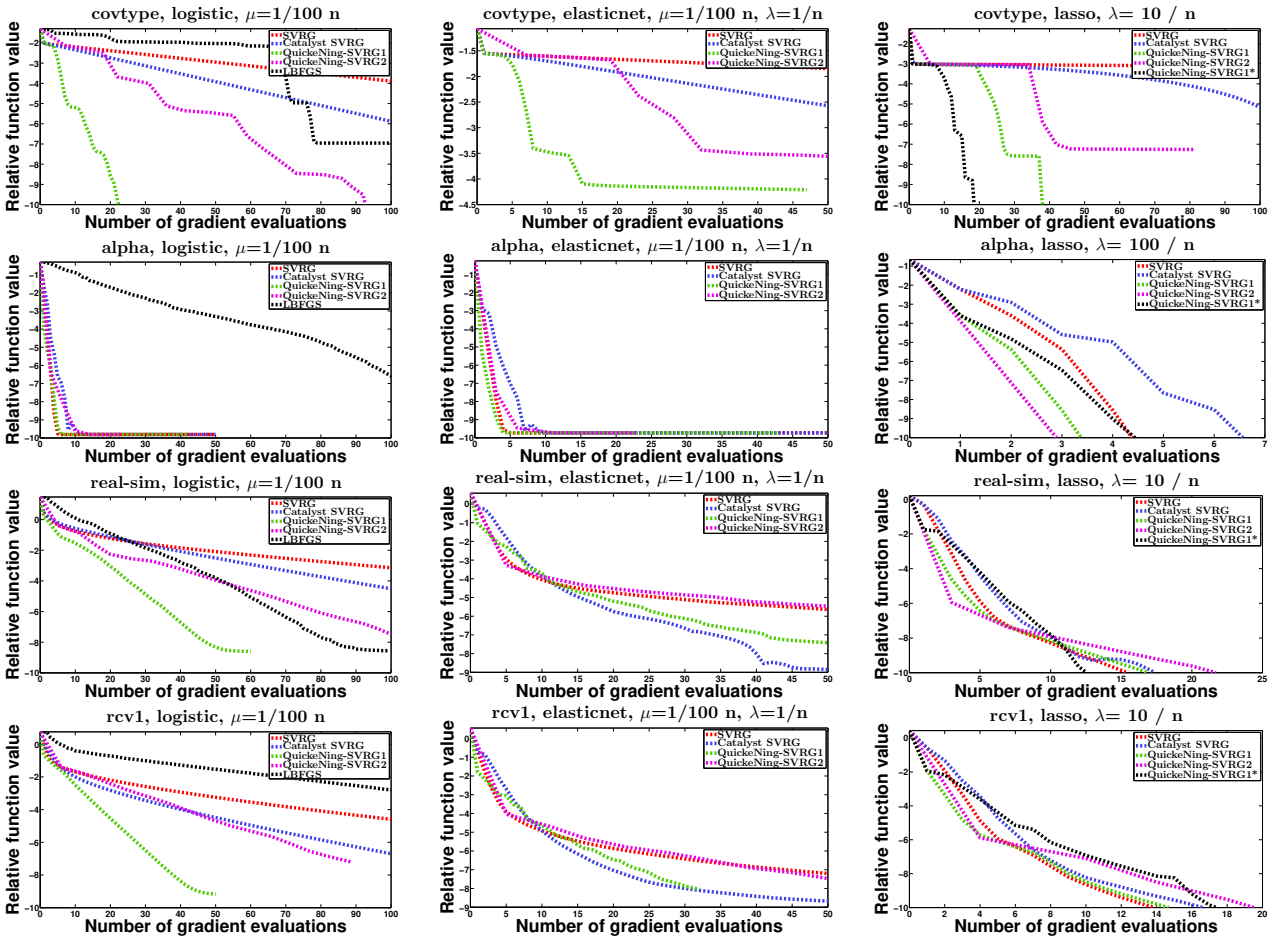

Figure 4.1 – Experimental study of the performance of QuickeNing-SVRG for minimizing large sums of functions. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

- **QuickeNing-SVRG1** as detailed above. For the Lasso problem, we also evaluate this heuristic with the sufficient decrease condition used for strongly convex problems. We call this heuristic variant **QuickeNing-SVRG1***.
- **QuickeNing-SVRG2**, as detailed above.

The result of the comparison is presented in Figure 4.1 and leads to the conclusions below, showing that **QuickeNing-SVRG1** is a safe heuristic, which never decreases the speed of the method **SVRG**:

- **L-BFGS** 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).

- **QuickeNing-SVRG1** is significantly faster than or on par with **SVRG** and **QuickeNing-SVRG2**.
- **QuickeNing-SVRG2** is significantly faster than, or on par with, or only slightly slower than **SVRG**.
- **QuickeNing-SVRG1** is significantly faster, or on par with **Catalyst-SVRG** except in two cases where **Catalyst-SVRG** performs slightly better.
- there is no clear conclusion regarding the performance of **QuickeNing-SVRG1*** vs **QuickeNing-SVRG1** for the Lasso problem.

4.5.4 QuickeNing-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 QuickeNing when applied to a basic proximal gradient descent algorithm such as ISTA. Specifically, we now consider

- **ISTA**: the classical proximal gradient descent algorithm ISTA [Beck and Teboulle, 2009] with back-tracking line-search to automatically adjust the Lipschitz constant of the gradient objective;
- **FISTA**: the accelerated variant of ISTA from [Beck and Teboulle, 2009].
- **L-BFGS**, **QuickeNing-ISTA1**, **QuickeNing-ISTA2**, and **QuickeNing-ISTA1***, as in the previous section when replacing SVRG by ISTA.

The results are reported in Figure 4.2 and lead to the following conclusions

- **L-BFGS** is slightly better on average than **QuickeNing-ISTA1** for smooth problems, which is not surprising since we use a state-of-the-art implementation with a well-calibrated line search.
- **QuickeNing-ISTA1** is always significantly faster than, or on par with **QuickeNing-ISTA2**.
- The **QuickeNing-ISTA** approaches are significantly faster than **FISTA** in 15 cases out of 16.
- for Lasso problems, **QuickeNing-ISTA1*** is sometimes faster than **QuickeNing-ISTA1**.

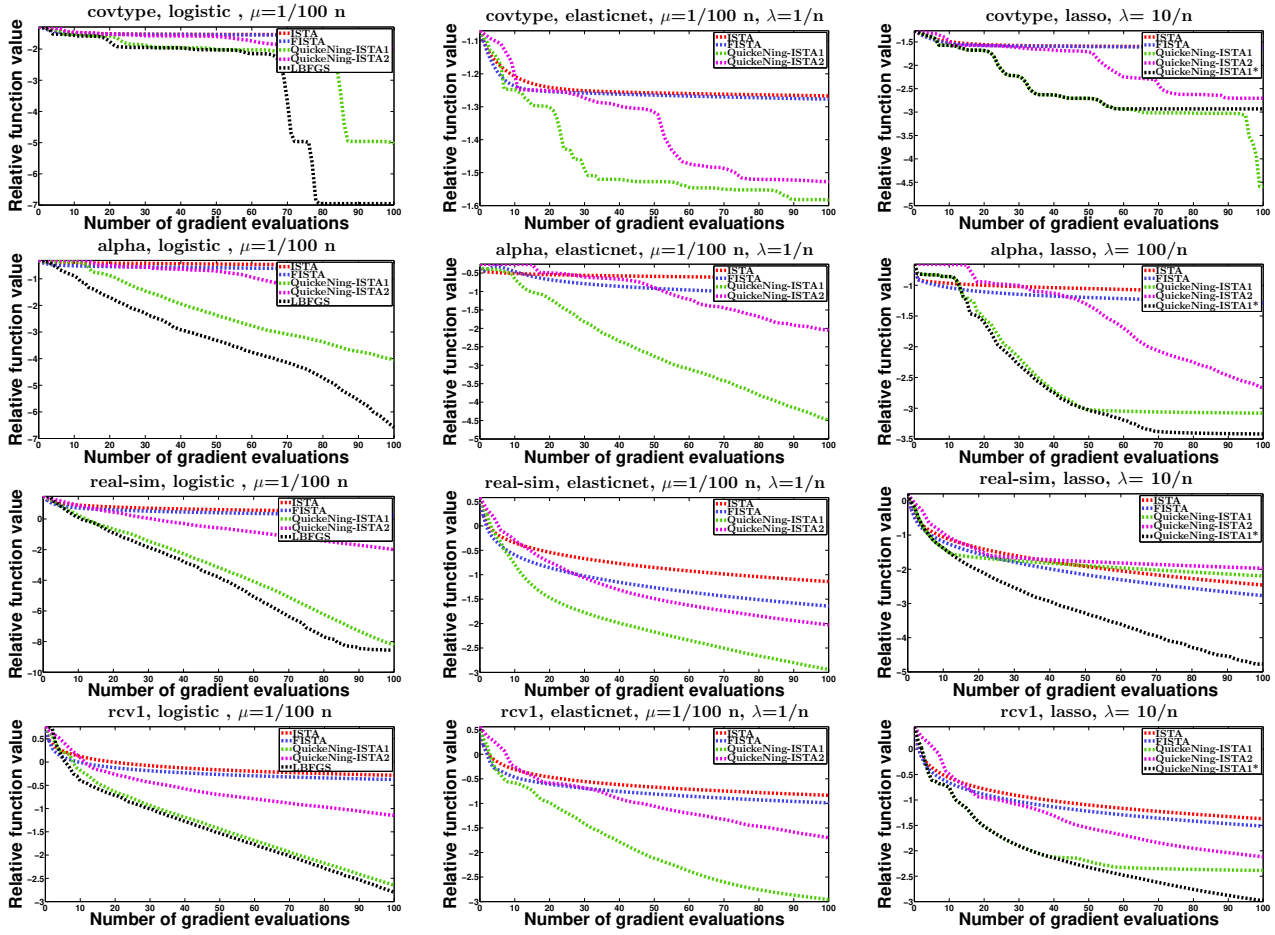

Figure 4.2 – Experimental study of the performance of QuickeNing-ISTA. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

4.5.5 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 QuickeNing. More precisely, we start with the parameter l and try the method **QuickeNing-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 4.3. 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 $l = 5$ seems to be a reasonable choice in practice.

The next experiment consists of studying the robustness of QuickeNing to the smoothing parameter κ. We present in Figure 4.4 an experiments by trying the values $\kappa = 10^i \kappa_0$, for

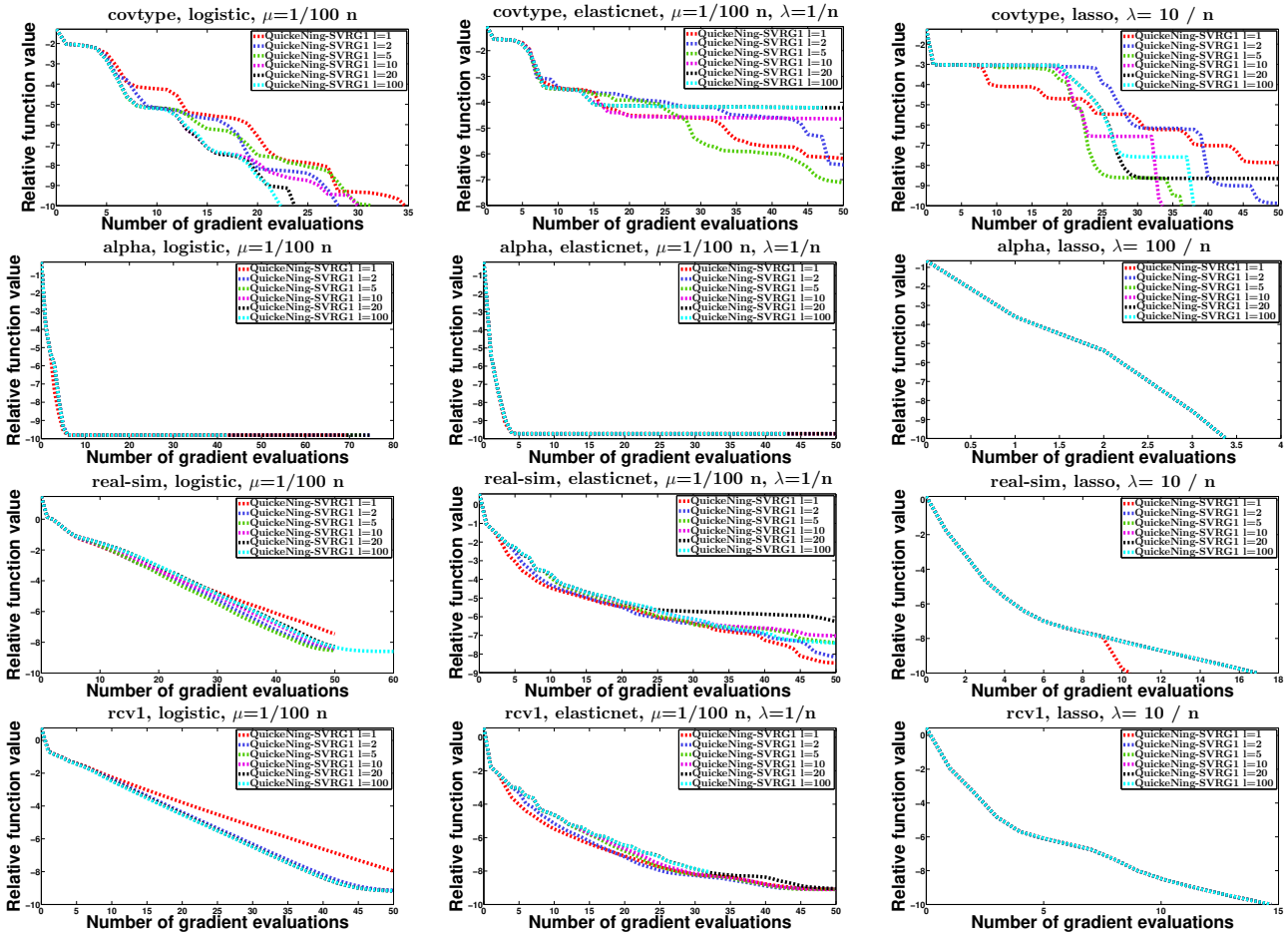

Figure 4.3 – Experimental study of influence of the limited-memory parameter l for QuickeNing-SVRG1. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

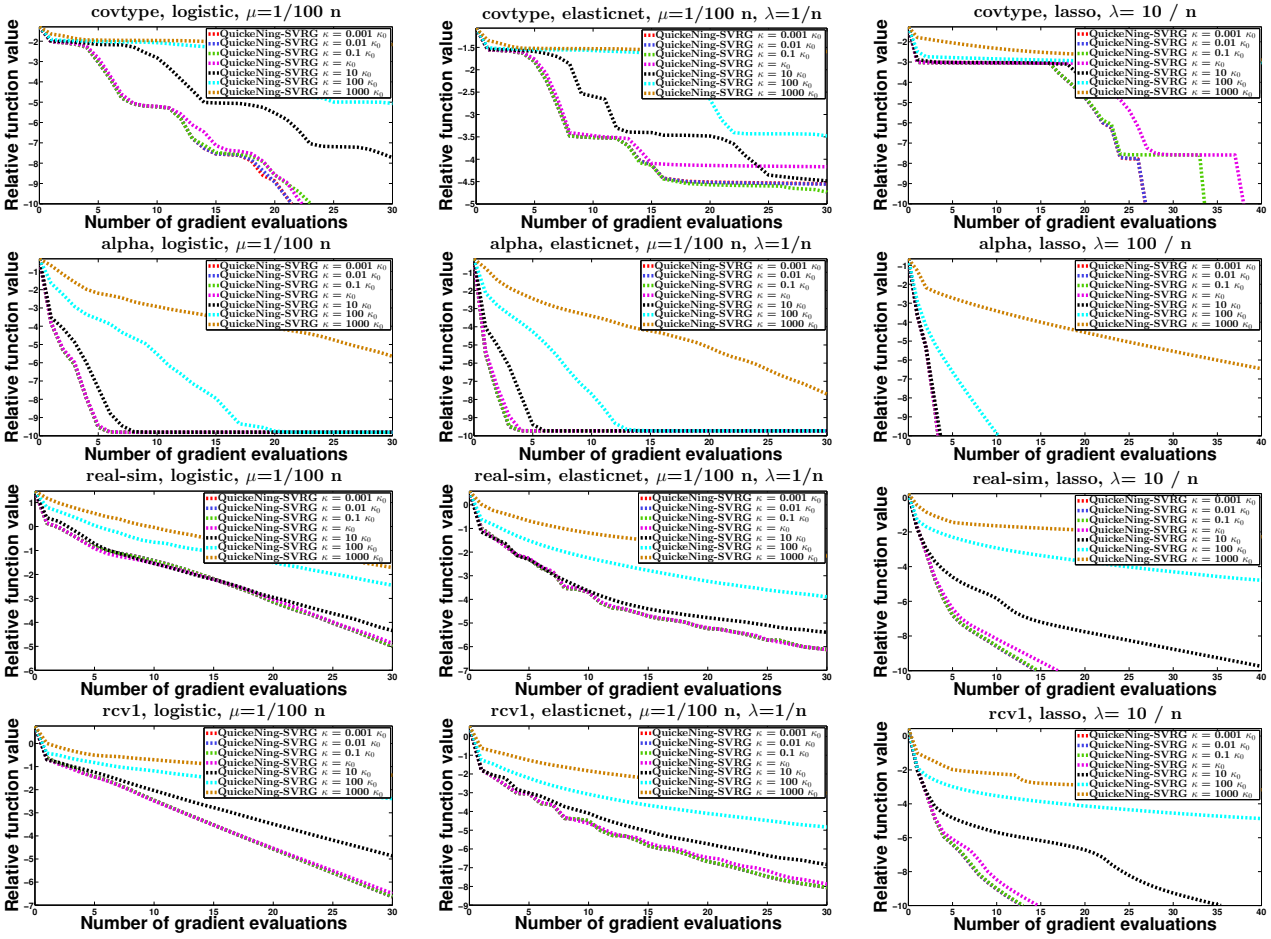

Figure 4.4 – Experimental study of influence of the smoothing parameter κ for QuickeNing-SVRG1. κ_0 denotes the default choice used in the previous experiments. We plot the value $F(x_k)/F^* - 1$ as a function of the number of gradient evaluations, on a logarithmic scale; the optimal value F^* is estimated with a duality gap.

$i = -3, -2, \ldots, 2, 3$, where κ_0 is the default parameter that we used in the previous experiments. The conclusion is clear: QuickeNing 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 in one experiment).

4.6 Discussions and concluding remarks

A few questions naturally arise regarding the QuickeNing 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.

4.6.1 Discussion of convergence rates

In this chapter, we have established the linear convergence of QuickeNing for strongly convex objectives when sub-problems are solved with enough accuracy. Since QuickeNing uses Quasi-Newton steps, one might have expected a superlinear convergence rate as several Quasi-Newton algorithms often enjoy [Byrd et al., 1987]. The situation is as follows. Consider the BFGS Quasi-Newton algorithm (without limited memory). As shown in [Chen and Fukushima, 1999], if the sequence $(\varepsilon_k)_{k\geq 0}$ decreases super-linearly, then, it is possible to design a scheme similar to QuickeNing that indeed enjoys a super-linear convergence rate. There are two major downsides though. The scheme of [Chen and Fukushima, 1999] 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 QuickeNing algorithm. Indeed, the convergence rate of QuickeNing scales linearly in the condition number μ_F/L_F, which suggests that a faster convergence rate could be obtained using a Nesterov-type acceleration scheme. Empirically, we did not observe any benefit of such a strategy, probably because of the pessimistic nature of the convergence rates that are typically obtained for Quasi-Newton approaches based on L-BFGS. Obtaining a linear convergence rate for an L-BFGS algorithm is still an important sanity check, but to the best of our knowledge, the gap in performance between these worst-case rates and practice has always been huge for this class of algorithms.

4.6.2 Other types of smoothing

Algorithm 6 (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 [Beck and Teboulle, 2012], whose family also encompasses the so-called Nesterov smoothing [Beck and Teboulle, 2012]. Other ways to smooth a function include randomization techniques [Duchi et al., 2012] 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 seen 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 [Beck and Teboulle, 2012], this property is also shared by other types of infimal convolutions. Therefore, QuickeNing could potentially be extended to such types of smoothing in place of the

Moreau-Yosida regularization. A major advantage of our approach, though, is its outstanding simplicity.

4.6.3 Concluding remarks

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

Chapter 5

Catalyst for non-convex optimization

Chapter abstract:
We introduce a generic scheme to solve nonconvex optimization problems using gradient-based algorithms originally designed for minimizing convex functions. When the objective is convex, the proposed approach enjoys the same properties as the Catalyst approach of Lin et al. [2015]. When the objective is nonconvex, it achieves the best known convergence rate to stationary points for first-order methods. Specifically, the proposed algorithm does not require knowledge about the convexity of the objective; yet, it obtains an overall worst-case efficiency of $\tilde{O}(\varepsilon^{-2})$ and, if the function is convex, the complexity reduces to the near-optimal rate $\tilde{O}(\varepsilon^{-2/3})$. We conclude the chapter by showing promising experimental results obtained by applying the proposed approach to SVRG and SAGA for sparse matrix factorization and for learning neural networks.

The material of this chapter is a short version of the following technical report, which is a joint work with Courtney Paquette and Dmitriy Drusvyatskiy from University of Washington and my PhD advisors:

C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui. Catalyst acceleration for gradient-based non-convex optimization. *arXiv preprint arXiv:1703.10993v2*, 2017

The paper is in preparation for submission to AISTATS 2018. For completeness, the technical report is presented in appendix D, which includes the proofs of all technical results of this chapter.

5.1 Introduction

We consider optimization problems of the form

$$\min_{x\in\mathbb{R}^p} \Big\{ f(x) := f_0(x) + \psi(x) \Big\}, \quad \text{where } f_0(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)\,. \tag{5.1}$$

Here each function $f_i\colon \mathbb{R}^p \to \mathbb{R}$ is smooth, the regularization $\psi\colon \mathbb{R}^p \to \bar{\mathbb{R}}$ may be nonsmooth, and $\bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. By considering extended-real-valued functions, this composite setting also encompasses constrained minimization by letting ψ be the indicator function of the constraints on x. Minimization of a regularized empirical risk objective of form (5.1) is central in machine learning. Whereas a significant amount of work has been devoted to this setting for convex problems, leading in particular to fast incremental algorithms [see, *e.g.*, Defazio et al., 2014a, Lan and Zhou, 2015, Mairal, 2015, Schmidt et al., 2017, Woodworth and Srebro, 2016, Xiao and Zhang, 2014], the question of minimizing efficiently (5.1) when the functions f_i and ψ may be nonconvex is still largely open today.

Yet, nonconvex problems in machine learning are of high interest. For instance, the variable x may represent the parameters of a neural network, where each term $f_i(x)$ measures the fit between x and a data point indexed by i, or (5.1) may correspond to a nonconvex matrix factorization problem (see Section 5.6). Besides, even when the data-fitting functions f_i are convex, it is also typical to consider nonconvex regularization functions ψ, for example for feature selection in signal processing [Hastie et al., 2015.]. In this work, we address two questions from nonconvex optimization:

1. How to apply a method for convex optimization to a nonconvex problem?
2. How to design an algorithm which does not need to know whether the objective function is convex while obtaining the optimal convergence guarantee if the function is convex?

Several pioneering works attempted to transfer ideas from the convex world to the nonconvex one, see, *e.g.*, [Ghadimi and Lan, 2016, Ghadimi et al., 2015]. Our work has a similar goal and studies the extension of Nesterov's acceleration for convex problems [Nesterov, 1983] to nonconvex composite ones. For nonconvex and L-smooth problems, gradient descent is optimal among first-order methods in terms of information-based complexity to find an ε-stationary point [Carmon et al., 2017][Thm. 2 Sec. 5]. Without additional assumptions on the function f, worst case complexity for first-order methods can not achieve better than $\mathcal{O}(\varepsilon^{-2})$ oracle queries [Cartis et al., 2010, 2014]. Under a stronger assumption that the objective function is C^2-smooth, state-of-the-art methods [*e.g.*, Carmon et al., 2016, 2017] only achieve marginal gain with complexity $O(\varepsilon^{-7/4}\log(1/\varepsilon))$, and do not appear to generalize to composite or stochastic settings. For this reason, our work fits within a broader stream of recent research on methods that *do not perform worse than gradient descent in the nonconvex case* (in terms of worst-case complexity), while *automatically accelerating for minimizing convex functions*. The hope is to see acceleration in practice for non-convex problems, by exploiting "hidden" convexity in the objective (*e.g.*, local convexity near the optimum, or convexity along the trajectory of iterates).

Our main contribution is a *generic* meta-algorithm, dubbed 4WD-Catalyst, which is able to use an optimization method $\mathcal{M}$, originally designed for convex problems, and turn it into an accelerated scheme that also applies to nonconvex objective functions. The proposed 4WD-Catalyst can be seen as a **4-W**heel-**D**rive extension of Catalyst [Lin et al., 2015] to all optimization "terrains". Specifically, without knowing whether the objective function is convex or not, our algorithm may take a method $\mathcal{M}$ designed for convex optimization problems with the same structure as (5.1), *e.g.*, SAGA [Defazio et al., 2014a], SVRG [Xiao and Zhang, 2014], and apply $\mathcal{M}$ to a sequence of sub-problems such that it provides a stationary point of the nonconvex objective. Overall, the number of iterations of $\mathcal{M}$ to obtain a gradient norm smaller than ε is $\tilde{O}(\varepsilon^{-2})$ in the worst case, while automatically reducing to $\tilde{O}(\varepsilon^{-2/3})$ if the function is convex.[1]

Related work. Inspired by Nesterov's fast gradient method for convex optimization [Nesterov, 2004], the first accelerated method performing universally well for nonconvex and convex problems was introduced in [Ghadimi and Lan, 2016]. Specifically, [Ghadimi and Lan, 2016] addresses composite problems such as (5.1) with $n=1$, and, provided the iterates are bounded, performs no worse than gradient descent on nonconvex instances with complexity $O(\varepsilon^{-2})$ on the gradient norm. When the problem is convex, it accelerates with complexity $O(\varepsilon^{-2/3})$. Extensions to Gauss-Newton methods were also recently developed in [Drusvyatskiy and Paquette, 2016]. To the best of our knowledge, convergence guarantees for accelerated gradient methods directly applied to nonconvex problem are unknown; however their performance escaping saddle points faster than gradient descent has been observed [O'Neill and Wright, 2017, Jin et al., 2017].

In [Li and Lin, 2015], a similar strategy is proposed, focusing instead on convergence guarantees under the so-called Kurdyka-Łojasiewicz inequality—a property corresponding to polynomial-like growth of the function, as shown by [Bolte et al., 2016]. Our scheme is in the same spirit as these previous papers, since it monotonically interlaces proximal-point steps (instead of proximal-gradient as in [Ghadimi et al., 2015]) and extrapolation/acceleration steps. A fundamental difference is that our method is generic and accommodates inexact computations, since we allow the subproblems to be approximately solved by any method we wish to accelerate.

By considering C^2-smooth nonconvex objective functions f with Lipschitz continuous gradient ∇f and Hessian $\nabla^2 f$, the authors of [Carmon et al., 2016] propose an algorithm with complexity $O(\varepsilon^{-7/4}\log(1/\varepsilon))$, based on iteratively solving convex subproblems closely related to the original problem. It is not clear if the complexity of their algorithm improves in the convex setting. Note also that the algorithm proposed in [Carmon et al., 2016] is inherently for C^2-smooth minimization. This implies that the scheme does not allow incorporating nonsmooth regularizers and cannot exploit finite sum structure.

Finally, a method related to SVRG [Johnson and Zhang, 2013] for minimizing large sums, while automatically adapting to the weak convexity constant of the objective function, is pro-

[1] In this section, the notation $\tilde{O}$ only displays the polynomial dependency with respect to ε for simplicity.

posed in [Allen-Zhu, 2017]. When the weak convexity constant is small (*i.e.*, the function is nearly convex), the proposed method enjoys an improved efficiency estimate. This algorithm, however, does not automatically accelerate for convex problems, in the sense that the rate is slower than $O(\varepsilon^{-3/2})$ in terms of target accuracy ε on the gradient norm.

Organization of the chapter. Section 5.2 presents mathematical tools for non-convex and non-smooth analysis. Section 5.3 introduces our algorithm, while Section 5.5 presents global convergence guarantees of the scheme and convergence guarantees when the algorithm wraps specific algorithms such as SAGA and SVRG. Section 5.6 is devoted to experiments on neural networks and matrix factorization.

5.2 Tools for nonconvex and nonsmooth optimization

Convergence results for nonsmooth optimization typically rely on the concept of subdifferential, which does not admit a unique definition in a nonconvex context [Borwein and Lewis, 2010]. In this chapter, we circumvent this issue by focusing on a broad class of nonconvex functions known as *weakly convex* or lower C^2 functions, for which all these constructions coincide. Weakly convex functions cover most of the interesting cases of interest in machine learning and resemble convex functions in many aspects.

Definition 5 (Weak convexity)**.** A function $f\colon \mathbb{R}^p \to \overline{\mathbb{R}}$ is $\rho-$*weakly convex* if for any points $x, y \in \mathbb{R}^p$ and $\lambda \in [0,1]$, the approximate secant inequality holds:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) + \frac{\rho\lambda(1-\lambda)}{2} \|x-y\|^2 .$$

Notice that ρ-weak convexity with $\rho = 0$ is exactly the definition of a convex function and it is easy to show that f is ρ-weakly convex if and only if the function $x \mapsto f(x) + \frac{\rho}{2}\|x\|^2$ is convex. In particular, a C^1-smooth function f is ρ-weakly convex if the gradient ∇f is ρ-Lipschitz, while a C^2-smooth function f is ρ-weakly convex if and only if $\nabla^2 f(x) \succeq -\rho I$ for all x. This resembles an equivalent condition for C^2-smooth and μ-strongly convex functions, namely $\nabla^2 f(x) \succeq \mu I$.

Useful characterizations of ρ-weakly convex functions rely on differential properties. Since the functions we consider here are nonsmooth, we use a generalized derivative construction. We mostly follow the standard monograph on the subject by Rockafellar and Wets [Rockafellar and Wets, 1998].

Definition 6 (Subdifferential)**.** Consider a function $f\colon \mathbb{R}^p \to \overline{\mathbb{R}}$ and a point x with $f(x)$ finite. The *subdifferential* of f at x is the set

$$\partial f(x) := \{v \in \mathbb{R}^p : f(y) \geq f(x) + v^T(y-x) + o(\|y-x\|) \quad \forall y \in \mathbb{R}^p\}.$$

Thus, a vector v lies in $\partial f(x)$ whenever the linear function $y \mapsto f(x) + v^T(y - x)$ is a lower-model of f, up to first-order around x. In particular, the subdifferential $\partial f(x)$ of a differentiable function f is the singleton $\{\nabla f(x)\}$, while for a convex function f it coincides with the subdifferential in the sense of convex analysis [see Rockafellar and Wets, 1998, Exercise 8.8]. It is useful to keep in mind that the sum rule, $\partial(f+g)(x) = \partial f(x) + \nabla g(x)$, holds for any differentiable function g.

In non-convex optimization, standard complexity bounds are derived to guarantee

$$\operatorname{dist}\Big(0, \partial f(x)\Big) \leq \varepsilon \ .$$

When $\varepsilon = 0$, we are at a stationary point and first-order optimality conditions are satisfied; that is, the directional derivative at the point is nonnegative in every direction. For functions that are nonconvex, first-order methods search for points with small subgradients, which does not necessarily imply small function values, in contrast to convex functions where the two criteria are much closer related.

5.3 The 4WD-Catalyst algorithm

We present here our main algorithm called 4WD-Catalyst. The proposed approach extends the Catalyst method [Lin et al., 2015] to potentially nonconvex problems, while enjoying the two following properties:

1. When the problem is non convex, the algorithm automatically adapts to the unknown weak convexity constant ρ.
2. When the problem is convex, the algorithm automatically accelerates in the sense of Nesterov, providing near-optimal convergence rates for first-order methods.

Main goal. As in the regular Catalyst algorithm of [Lin et al., 2015], the proposed scheme wraps in an outer loop a minimization algorithm $\mathcal{M}$ used in an inner loop. The goal is to leverage a method $\mathcal{M}$ that is able to exploit the problem structure (finite-sum, composite) in the convex case, and benefit from this feature when dealing with a new problem with unknown convexity; remarkably, $\mathcal{M}$ does not need to have any convergence guarantee for nonconvex problems to be used in 4WD-Catalyst.

Two-step subproblems. In each iteration, 4WD-Catalyst forms subproblems of the form

$$\min_x \quad f_\kappa(x; y) := f(x) + \frac{\kappa}{2}\|x - y\|^2. \qquad (\mathcal{P})$$

We call y the *prox-center* of the subproblem and any minimizer a *proximal point*. The perturbed function $f_\kappa(x; y)$ satisfies the important property: $f_\kappa(\cdot; y)$ is $(\kappa - \rho)$-strongly convex for any $\kappa > \rho$. The addition of the quadratic to f makes the subproblem more "convex". That is,

when f is non convex, a large enough κ yields convex subproblems; even when f is convex, the quadratic perturbation nonetheless improves conditioning.

We now describe the k'th iteration of Algorithm 8. To this end, suppose we have available iterates x_{k-1} and v_{k-1}. At the center of our Algorithm 8 are two main sequences of iterates $(\bar{x}_k)_k$ and $(\tilde{x}_k)_k$, obtained from approximately solving two subproblems of the form $\mathcal{P}$.

1. **P**roximal point step. We first perform an inexact proximal point step with prox-center x_{k-1}:

$$\bar{x}_k \approx \arg\min_x f_\kappa(x; x_{k-1}) \quad \text{[Proximal-point step]} \tag{5.2}$$

2. **A**ccelerated proximal point step. Then we build the next prox-center y_k as the combination

$$y_k = \alpha_k v_{k-1} + (1 - \alpha_k) x_{k-1}. \tag{5.3}$$

Next we use y_k as a prox-center and update the next extrapolation term:

$$\tilde{x}_k \approx \arg\min_x f_\kappa(x; y_k) \qquad \text{[Accelerated proximal-point step]} \tag{5.4}$$

$$v_k = x_{k-1} + \tfrac{1}{\alpha_k}(\tilde{x}_k - x_{k-1}) \qquad \text{[Extrapolation]} \tag{5.5}$$

where $\alpha_{k+1} \in (0, 1)$ is a sequence of coefficients satisfying $(1 - \alpha_{k+1})/\alpha_{k+1}^2 = 1/\alpha_k^2$. Essentially, the sequences $(\alpha_k)_k, (y_k)_k, (v_k)_k$ are built upon the extrapolation principles of [Nesterov, 2004].

Picking the best. At the end of iteration k, we have two iterates, resp. $\bar{x}_k$ and $\tilde{x}_k$. Following [Ghadimi and Lan, 2016], we simply choose the best of the two in terms of their objective values, that is we choose x_k such that

$$f(x_k) \leq \min\left\{f(\bar{x}_k), f(\tilde{x}_k)\right\}.$$

The proposed scheme blends the two steps in a synergistic way, allowing us to recover the near-optimal rates of convergence in both worlds: convex and non-convex. Intuitively, when $\bar{x}_k$ is chosen, it means that Nesterov's extrapolation step "fails" to accelerate convergence.

We present now our strategy to set the parameters of 4WD-Catalyst so that the resulting algorithm a) automatically adapts to the unknown weak convexity constant ρ; b) enjoys a near-optimal rate of convergence in both convex and non-convex settings.

5.4 Optimal parameters and adaptation

When κ is large enough, the subproblems become strongly convex; thus globally solvable. Henceforth, we will assume that $\mathcal{M}$ satisfies the following natural linear convergence assumption.

Algorithm 7 Auto-adapt (x, κ, T)

input $x \in \mathbb{R}^p$, method $\mathcal{M}$, $\kappa > 0$, number of iterations T.

Repeat Run T iterations of $\mathcal{M}$ initializing from $z_0 = x$, or $z_0 = \operatorname{prox}_{\eta\psi}(x - \eta\nabla f_0(x; y))$ with $\eta = 1/(L + \kappa)$ for composite objectives $f = f_0 + \psi$ with L-smooth function f_0, to obtain

$$z_T \approx \arg\min_{z \in \mathbb{R}^p} f_\kappa(z; x).$$

If $f_\kappa(z_T; x) \le f_\kappa(x; x)$ and $\operatorname{dist}(0, \partial f_\kappa(z_T; x)) \le \kappa \|z_T - x\|$,
then go to output.
else repeat with $\kappa \to 2\kappa$.

output (z_T, κ).

Algorithm 8 4WD-Catalyst

input Fix a point $x_0 \in \operatorname{dom} f$, real numbers $\kappa_0, \kappa_{\text{cvx}} > 0$ and $T, S > 0$, and an opt. method $\mathcal{M}$.

initialization: $\alpha_1 = 1$, $v_0 = x_0$.

repeat for $k = 1, 2, \ldots$

1. Compute
$$(\bar{x}_k, \kappa_k) = \text{Auto-adapt } (x_{k-1}, \kappa_{k-1}, T).$$
2. Compute $y_k = \alpha_k v_{k-1} + (1 - \alpha_k) x_{k-1}$ and apply $S \log(k + 1)$ iterations of $\mathcal{M}$ to find
$$\tilde{x}_k \approx \arg\min_{x \in \mathbb{R}^p} f_{\kappa_{\text{cvx}}}(x, y_k). \tag{5.6}$$
3. Update v_k and α_{k+1} by
$$v_k = x_{k-1} + \tfrac{1}{\alpha_k}(\tilde{x}_k - x_{k-1}) \quad \text{and} \quad \alpha_{k+1} = \frac{\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2}{2}.$$
4. Choose x_k to be any point satisfying $f(x_k) = \min\{f(\bar{x}_k), f(\tilde{x}_k)\}$.

until the stopping criterion $\operatorname{dist}\Big(0, \partial f(\bar{x}_k)\Big) < \varepsilon$

Linear convergence of $\mathcal{M}$ for strongly-convex problems. We assume that for any $\kappa > \rho$, there exist $A_\kappa \ge 0$ and $\tau_\kappa \in (0, 1)$ so that the following hold:

1. For any prox-center $y \in \mathbb{R}^p$, define $f_\kappa^*(y) = \min_z f_\kappa(z, y)$. For any initial point $z_0 \in \mathbb{R}^p$, the iterates $\{z_t\}_{t \ge 1}$ generated by $\mathcal{M}$ on the problem $\min_z f_\kappa(z; y)$ satisfy
$$\operatorname{dist}^2(0, \partial f_\kappa(z_t; y)) \le A_\kappa (1 - \tau_\kappa)^t \Big(f_\kappa(z_0; y) - f_\kappa^*(y) \Big). \tag{5.7}$$

2. The rates τ_κ and constants A_κ are increasing in κ.

When the function is strongly convex, the first condition is equivalent to the error $f_\kappa(z_k; y) - f_\kappa^*(y)$ decreasing geometrically to zero (convergence in sub-gradient norm and function values are equivalent in this case). If the method is randomized, we allow (5.7) to hold in expectation; see Sec. 4. in Appendix. All algorithms of interest, (*e.g.*, gradient descent, SVRG, SAGA) satisfy these properties.

Adaptation to weak convexity and predefining T. Recall that we add a quadratic to f to make each subproblem convex. Thus, we should set $\kappa > \rho$, if ρ were known. On the other other hand, we do not want κ too large, as that may slow down the overall algorithm. In any case, it is difficult to have an accurate estimate of ρ for machine learning problems such as neural networks. Thus, we propose a procedure described in Algorithm 7 to automatically adapt to ρ.

The idea is to fix in advance a number of iterations T, let $\mathcal{M}$ run on the subproblem for T iterations, output the point z_T, and check if a sufficient decrease occurs. We show that if we set $T = \tilde{O}(\tau_L^{-1})$, where $\tilde{O}$ hides logarithmic dependencies in L and A_L, where L is the Lipschitz constant of the smooth part of f; then, if the subproblem were convex, the following conditions would be guaranteed:

1. Descent condition: $f_\kappa(z_T; x) \leq f_\kappa(x; x)$;
2. Adaptive stationary condition: $\operatorname{dist}\Big(0, \partial f_\kappa(z_T; x)\Big) \leq \kappa \left\| z_T - x \right\|$.

Thus, if either condition is not satisfied, then the subproblem is deemed not convex and we double κ and repeat. The procedure yields an estimate of ρ in a logarithmic number of increases; see lem. D.3.

The *descent condition* is a sanity check, which ensures the iterates generated by the algorithm always decrease the function value. Without it, the stationarity condition alone is insufficient because of the existence of local maxima in nonconvex problems.

The *adaptive stationarity property* controls the inexactness of the subproblem in terms of subgradient norm. In a non convex setting, the subgradient norm is convenient, since we cannot access $f_\kappa(z_T, x) - f_\kappa^*(x)$. Furthermore, unlike the stationary condition $\operatorname{dist}\Big(0, \partial f_\kappa(z_T; x)\Big) < \varepsilon$, where an accuracy ε is predefined, the adaptive stationarity condition depends on the iterate z_T. This turns out to be essential in deriving the global complexity. Sec. 3 in the Appendix contains more details.

Relative stationarity and predefining S. One of the main differences of our approach with the Catalyst algorithm of [Lin et al., 2015] is to use a *pre-defined* number of iterations, T and S, for solving the subproblems. We introduce κ_{cvx}, a $\mathcal{M}$ dependent smoothing parameter and set it in the same way as the smoothing parameter in [Lin et al., 2015]. The automatic acceleration of our algorithm when the problem is convex is due to extrapolation steps in Step 2-3 of Algo. 8. We show that if we set $S = \tilde{O}\left(\tau_{\kappa_{\text{cvx}}}^{-1}\right)$, where $\tilde{O}$ hides logarithmic dependencies

in L, κ_κ, and $A_{\kappa_{\text{cvx}}}$, then we can be sure that, for convex,

$$\operatorname{dist}\Big(0, \partial f_{\kappa_{\text{cvx}}}(\tilde{x}_k; y_k)\Big) < \frac{\kappa_{\text{cvx}}}{k+1} \|\tilde{x}_k - y_k\| . \tag{5.8}$$

This relative stationarity of $\tilde{x}_k$, including the choice of κ_{cvx}, shall be crucial to guarantee that the scheme accelerates in the convex setting. An additional $k+1$ factor appears compared to the previous adaptive stationary condition because we need higher accuracy for solving the subproblem to achieve the accelerated rate in $1/\sqrt{\varepsilon}$. Therefore, an extra $\log(k+1)$ factor of iterations is needed; see Sec. 3 and Sec. 4 in the Appendix.

We shall see, in Sec. 5.6, that our strategy of predefining T and S works quite well. The theoretical bounds we derive are, in general, too conservative; we observe in our experiments that one may choose T and S significantly smaller than the theory suggests and still retain the stopping criteria.

5.5 Global convergence and applications to existing algorithms

With the previous section in mind, we can now present the complexity of our algorithm, which takes into account the cost of approximately solving the subproblems (5.2) and (5.4).

Theorem 9 (Global complexity bounds for 4WD-Catalyst)**.** *Choose $T = \tilde{O}(\tau_L^{-1})$ and $S = \tilde{O}(\tau_{\kappa_{cvx}}^{-1})$ (see Theorem 4.5 in Appendix). Then the following are true.*

1. *Algorithm 8 generates a point x satisfying $\operatorname{dist}\Big(0, \partial f(x)\Big) \leq \varepsilon$ after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{cvx}}^{-1}\right) \cdot \frac{L(f(x_0) - f^*)}{\varepsilon^2}\right) \qquad \textit{iterations of the method } \mathcal{M}.$$

2. *If f is convex, Algorithm 8 generates a point x satisfying $\operatorname{dist}\Big(0, \partial f(x)\Big) \leq \varepsilon$ after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{cvx}}^{-1}\right) \cdot \frac{L^{1/3}\left(\kappa_{cvx}\|x^* - x_0\|^2\right)^{1/3}}{\varepsilon^{2/3}}\right) \qquad \textit{iterations of the method } \mathcal{M}.$$

3. *If f is convex, Algorithm 8 generates a point x satisfying $f(x) - f^* \leq \varepsilon$ after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{cvx}}^{-1}\right) \cdot \frac{\sqrt{\kappa_{cvx}\|x^* - x_0\|^2}}{\sqrt{\varepsilon}}\right) \qquad \textit{iterations of the method } \mathcal{M}.$$

Here $\tilde{O}$ hides universal constants and logarithmic dependencies in $A_{\kappa_{\text{cvx}}}$, A_L, κ_0, κ_{cvx}, ε, and $\|x^* - x_0\|^2$.

If $\mathcal{M}$ is a first order method, the convergence guarantee in the convex setting is *near-optimal*, up to logarithmic factors, when compared to $O(1/\sqrt{\varepsilon})$ [Lin et al., 2015, Woodworth and Srebro, 2016]. In the non-convex setting, our approach matches, up to logarithmic factors, the best known rate for this class of functions, namely $O(1/\varepsilon^2)$ [Cartis et al., 2010, 2014]. Moreover, our rates dependence on the dimension and Lipschitz constant equals, up to log factors, the best known dependencies in both the convex and nonconvex setting. These logarithmic factors may be the price we pay for having a generic algorithm.

Choice of κ_{cvx}. The parameter κ_{cvx} drives the convergence rate of 4WD-Catalyst in the convex setting. To determine κ_{cvx}, we compute the global complexity of our scheme as if $\rho = 0$, hence using the same reasoning as [Lin et al., 2015]. The rule consists in maximizing the ratio $\tau_\kappa/\sqrt{L+\kappa}$. Then, the choice of κ_0 is independent of $\mathcal{M}$; it is an initial lower estimate for the weak convexity constant, ρ. We provide a detailed derivation of all the variables for each of the considered algorithms in the appendix (Sec.5 in Appendix)

5.5.1 Applications

We now compare the guarantees obtained before and after applying 4WD-Catalyst to some specific optimization methods $\mathcal{M}$ (full gradient, SAGA, and SVRG). In the convex setting, the accuracy is stated in terms of optimization error, $f(x) - f^* \leq \varepsilon$ and in the nonconvex setting, in terms of stationarity condition $\text{dist}(0, \partial f(x)) < \varepsilon$.

Full gradient method. First, we consider the simplest case of applying our method to the full gradient method (FG). Here, the optimal choice for κ_{cvx} is L. In the convex setting, we get the accelerated rate $O(n\sqrt{L/\varepsilon}\log(1/\varepsilon))$ which is consistent with Nesterov's accelerated variant (AFG) up to logarithmic factors. In the nonconvex case, our approach achieves no worse rate than $O(nL/\varepsilon^2\log(1/\varepsilon))$, which is consistent with the standard gradient descent up to logarithmic factors. We note that under stronger assumptions, namely C^2-smoothness of the objective, the accelerated algorithm in [Carmon et al., 2017] achieves the same rate as (AFG) for the convex setting and $O(\varepsilon^{-7/4}\log(1/\varepsilon))$ for the nonconvex setting. Their approach, however, does not extend to composite setting nor to stochastic methods. Our marginal loss is the price we pay for considering a much larger class of functions.

Randomized incremental gradient. We now consider randomized incremental gradient methods such as SAGA [Defazio et al., 2014a] and (prox) SVRG [Xiao and Zhang, 2014]. Here, the optimal choice for κ_{cvx} is $O(L/n)$. Under the convex setting, we achieve an accelerated rate of $O(\sqrt{n}\sqrt{L/\varepsilon}\log(1/\varepsilon))$. Direct applications of SVRG and SAGA have no convergence guarantees in the non-convex setting. With our approach, the resulting algorithm matches the guarantees for FG up to log factors.

[2] See [Xiao and Zhang, 2014, Defazio et al., 2014a]

Table 5.1 – Comparison of rates of convergence, before and after the 4WD-Catalyst , resp. in the non-convex and convex cases. For the comparison, in the convex case, we only present the number of iterations to obtain a point x satisfying $f(x) - f^* < \varepsilon$. In the non-convex case, we show the number of iterations to obtain a point x satisfying $\text{dist}(0, \partial f(x)) < \varepsilon$.

	Stepsize	Nonconvex	Convex
FG	$O\left(\frac{1}{L}\right)$	$O\left(n\frac{L}{\varepsilon^2}\right)$	$O\left(n\frac{L}{\varepsilon}\right)$
4WD-Catalyst-FG	$O\left(\frac{1}{L}\right)$	$\tilde{O}\left(n\frac{L}{\varepsilon^2}\right)$	$\tilde{O}\left(n\sqrt{\frac{L}{\varepsilon}}\right)$

	Stepsize	Nonconvex	Convex
SVRG/SAGA[2]	$O\left(\frac{1}{L}\right)$	not avail.	$O\left(n\frac{L}{\varepsilon}\right)$
ncvx-SVRG/SAGA[3]	$O\left(\frac{1}{\sqrt{n}L}\right)$	$O\left(n^{2/3}\frac{L}{\varepsilon^2}\right)$	$O\left(\sqrt{n}\frac{L}{\varepsilon}\right)$
4WD-Catalyst-SVRG/SAGA	$O\left(\frac{1}{L}\right)$	$\tilde{O}\left(n\frac{L}{\varepsilon^2}\right)$	$\tilde{O}\left(\sqrt{n}\sqrt{\frac{L}{\varepsilon}}\right)$

5.6 Experiments

We investigate the performance of 4WD-Catalyst in two standard non-convex problems in machine learning. We report experimental results of 4WD-Catalyst when applied to two different algorithms: SVRG [Xiao and Zhang, 2014] and SAGA [Defazio et al., 2014a]. We compare the following algorithms:

- Nonconvex Prox-SVRG/SAGA [Reddi et al., 2016b]: stepsize $\eta = 1/Ln$
- Convex Prox-SVRG/SAGA [Xiao and Zhang, 2014, Defazio et al., 2014a]: stepsize $\eta = 1/2L$
- 4WD-Catalyst SVRG/SAGA: stepsize $\eta = 1/2L$

[3] See [Reddi et al., 2016a,b]. Here we present the convergence result without any mini-batching. In the mini-batch case, the same convergence can be obtained with a batch size $b = n^{2/3}$ and the large stepsize $O(1/L)$ is allowed.

Following the same experimental setting as [Reddi et al., 2016b], we evaluate algorithms in the absent of mini-batch. The original version of SVRG (resp. SAGA), convex SVRG (resp. SAGA), was designed for minimizing convex objectives. We report their results, while there is no theoretical guarantee on their behavior when venturing into nonconvex terrains. We also report the results of recently proposed variants, Nonconvex SVRG/SAGA, designed for minimizing nonconvex objectives. The proposed algorithms 4WD-Catalyst SVRG and 4WD-Catalyst SAGA enjoy the strong theoretical guarantees stated in Sec. 3.

Parameter settings We start from an initial estimate of the Lipschitz constant L and use the theoretically recommended $\kappa_0 = \kappa_{\text{cvx}} = 2L/n$. The number of inner iterations is to $T = S = n$ in all experiments, which boils down to making one pass at most over the data for solving each sub-problem. We simply drop the $\log(k)$ dependency while solving the subproblem in (5.6). These choices turn out to be justified *a posteriori*, as both SVRG and SAGA have a much better convergence rate in practice than the theoretical rate derived from a worst-case analysis. Indeed, in all experiments, one pass over the data to solve each sub-problem is enough to guarantee sufficient descent. We focus in the main text on the results for SVRG. We relegate results for SAGA and details about experiments to Sec.6 in Appendix.

Sparse matrix factorization a.k.a. dictionary learning. Dictionary learning consists of representing a dataset $X = [x_1, \cdots, x_n] \in \mathbb{R}^{m \times n}$ as a product $X \approx DA$, where D in $\mathbb{R}^{m \times p}$ is called a dictionary, and A in $\mathbb{R}^{p \times n}$ is a sparse matrix. The classical non-convex formulation [see Mairal et al., 2014] can be reformulated as the equivalent finite-sum problem $\min_{D \in \mathcal{C}} \frac{1}{n} \sum_{i=1}^{n} f_i(D)$ with

$$f_i(D) := \min_{\alpha \in \mathbb{R}^p} \frac{1}{2} \|x_i - D\alpha\|_2^2 + \psi(\alpha). \tag{5.9}$$

ψ is a sparsity-inducing regularization and $\mathcal{C}$ is chosen as the set of matrices whose columns are in the ℓ_2-ball; see Sec.6 in Appendix. We consider elastic-net regularization $\psi(\alpha) = \frac{\mu}{2}\|\alpha\|^2 + \lambda\|\alpha\|_1$ of [Zou and Hastie, 2005], which has a sparsity-inducing effect, and report the corresponding results in Figure 5.1, learning a dictionary in $\mathbb{R}^{m \times p}$ with $p = 256$ elements, on a set of whitened normalized image patches of size $m = 8 \times 8$. Parameters are standard ones in this literature [Mairal et al., 2014]—that is, a small value $\mu = 1e-5$, and $\lambda = 0.25$, leading to sparse matrices A (on average ≈ 4 non-zero coefficients per column of A).

Neural networks. We consider simple binary classification problems for learning neural networks. Assume that we are given a training set $\{a_i, b_i\}_{i=1}^n$, where the variables b_i in $\{-1, +1\}$ represent class labels, and a_i in $\mathbb{R}^p$ are feature vectors. The estimator of a label class is now given by a two-layer neural network $\hat{b} = \text{sign}(w_2^\top \sigma(W_1^\top a))$, where W_1 in $\mathbb{R}^{p \times d}$ represents the weights of a hidden layer with d neurons, w_2 in $\mathbb{R}^d$ carries the weight of the network's second layer, and $\sigma(u) = \log(1 + e^u)$ is a non-linear function, applied point-wise to its arguments. We

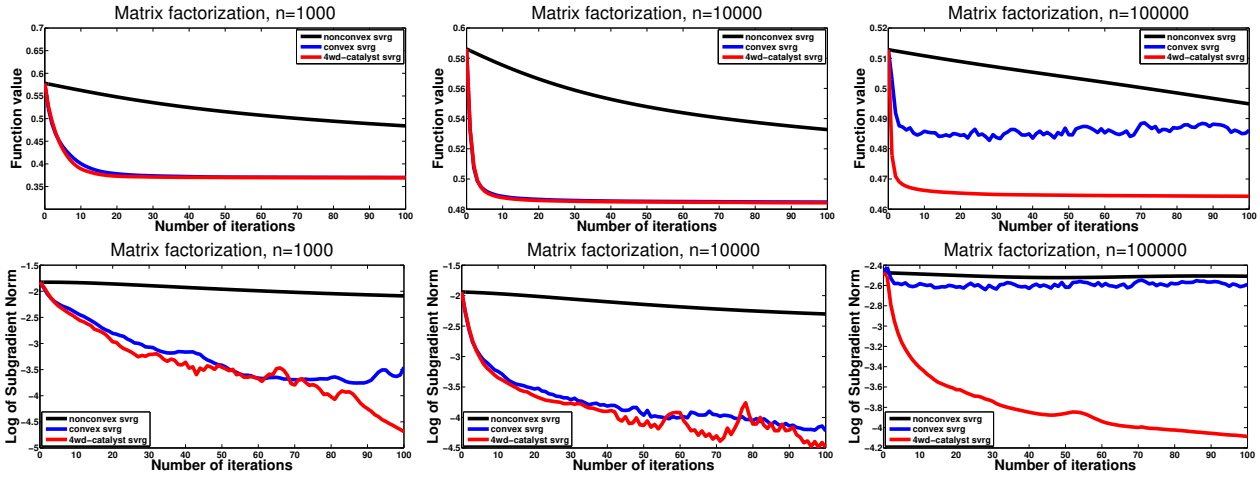

Figure 5.1 – Dictionary learning experiments. We plot the function value (top) and the subgradient norm (bottom). From left to right, we vary the size of the dataset from $n = 1\,000$ to $n = 100\,000$

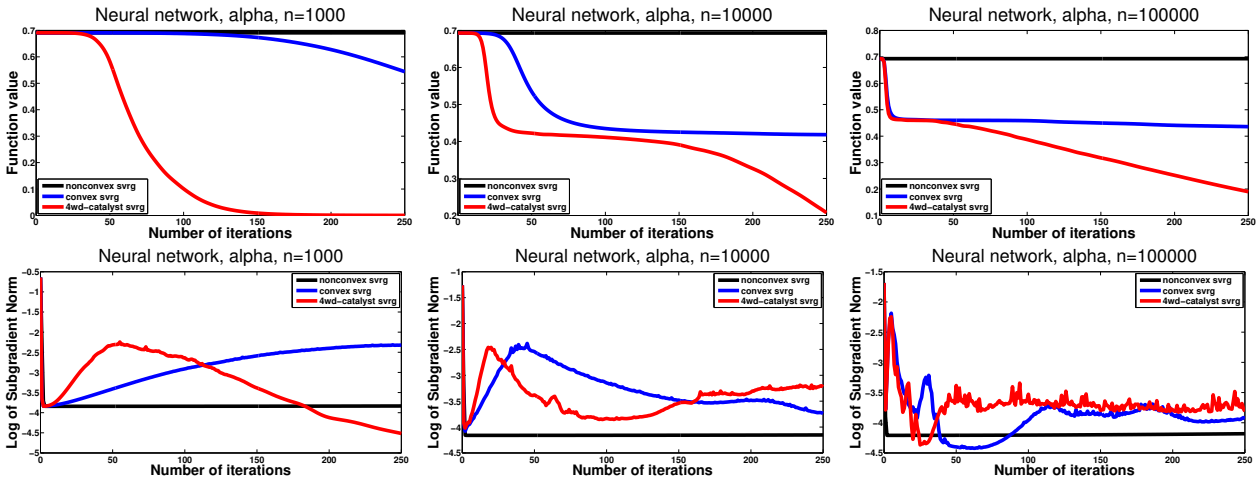

Figure 5.2 – Neural network experiments. Same experimental setup as in Fig. 5.1. From left to right, we vary the size of the dataset's subset from $n = 1\,000$ to $n = 100\,000$.

use the logistic loss to fit the estimators to the true labels and report experimental results on the two datasets alpha and covtype. The weights of the network are randomly initialized and we fix the number of hidden neurons to $d = 100$.

Computational cost. For the Convex-SVRG and Nonconvex-SVRG, one iteration corresponds to one pass over the data in the plots. On the one hand, since 4WD-Catalyst-SVRG solves two sub-problems per iteration, the cost per iteration is twice that of the Convex-SVRG and Nonconvex-SVRG. On the other hand, in the experiments, we observe that, everytime acceleration occurs, then $\tilde{x}_k$ is almost always preferred to $\bar{x}_k$ in step 4 of 4WD-Catalyst, half

of the computations are in fact not performed when running 4WD-Catalyst-SVRG.

Experimental conclusions. In matrix factorization experiments, we observe that 4WD-Catalyst-SVRG always outperforms the competing algorithms. Nonconvex-SVRG has slower convergence in objective values and Convex-SVRG is not always converging; see in particular right panel in Fig. 5.1. Therefore 4WD-Catalyst-SVRG offers a more stable option than Convex-SVRG for minimizing nonconvex objectives. Furthermore, in these experiments 4WD-Catalyst-SVRG enjoys a faster convergence in objective values. This confirms the remarkable ability of 4WD-Catalyst-SVRG to adapt to nonconvex terrains. Similar conclusions hold when applying 4WD-Catalyst to SAGA; see Sec.6. in Appendix, which demonstrate the genericity of 4WD-Catalyst.

In neural network experiments, we observe that 4WD-Catalyst-SVRG converges much faster in terms of objective values than the competing algorithms. Nonconvex-SVRG with the theoretically recommended sequence of step-sizes [Reddi et al., 2016b] compares unfavorably here, which implies that the recommended step-sizes are too pessimistic hence too small. We also observe an interesting phenomenon: the subgradient norm may increase at some point then decrease, while the function value keeps decreasing, as the algorithm proceeds. This suggests that the extrapolation step, or the Auto-adapt procedure, is helpful to escape bad stationary points, *e.g.*, saddle-points. We leave study of this particular phenomenon as a potential direction for future work.

Chapter 6

Discussions and concluding remarks

In this manuscript, we have introduced several generic acceleration schemes for accelerating gradient-based optimization methods. Our approaches are adapted to large-scale and high-dimensional convex problems a with composite finite sum structure. Moreover, one of our methods is extended to non convex formulations, which allows us to apply it to a large range of machine learning problems. In this section, we summarize our contributions and discuss possible directions for future work.

First, we have presented a proximal variant of the Finito/MISO algorithm. The algorithm we developed is able to deal with non smooth regularizations and it removes a constraint on the number of data points required in the original Finito/MISO algorithm. We have shown that our algorithm converges linearly with a similar rate as other variance reduction based incremental algorithms, when the objective function is strongly convex.

Second, we introduce a generic acceleration scheme called Catalyst. It provides acceleration in the sense of Nesterov to a large class of first-order optimization methods. We prove that for both convex and strongly convex problems, a near-optimal convergence rate, up to a logarithmic factor, is achieved by applying Catalyst. We experimentally demonstrate that the acceleration is useful, especially for ill-conditioned problems.

We then present another generic mechanism, QuickeNing, to accelerate existing first-order optimization algorithms with quasi-Newton principles. It is able to exploit the composite structure of the objective function and take into account the curvature information. To the best of our knowledge, QuickeNing is the first generic algorithm enjoying both of these features. We show that it provides significant improvement in practice over competing methods including the Catalyst acceleration. However, in contrast to Catalyst where a theoretical acceleration is guaranteed, QuickeNing does not improve the convergence rate in the worst case analysis. Thus a huge gap remains between the practical performance and the theoretical analysis, which is a common observation for quasi-Newton methods.

Finally, we extend Catalyst acceleration to non convex objective functions. The proposed algorithm does not require to know the function is convex or not and it automatically adapts to the convexity. When the objective is non convex, it converges to a stationary point with a

complexity not worse than the gradient descent method; when the objective is convex, acceleration will be obtained as the original Catalyst algorithm. We provide experiments on problems of matrix factorization and neural networks to show that our approach converges faster than existing algorithms.

Catalyst is our first attempt to the field of non convex optimization, it is possible to extend QuickeNing in a similar way which we leave to future work. The main difference between convex and non convex optimization is that when the function is convex, all stationary points are global minima; but when the function is non convex, a stationary point may not even be a local minimum. Thus, an important subject in non convex optimization is to study whether an optimization method can escape bad stationary points, *e.g.*, saddle-points. On one hand, positive results have been proven by incorporating Hessian information. Nesterov and Polyak [2006] shows that a cubic regularization of Newton method converges to second-order stationary points. Later, algorithms based on Hessian-vector products has been proposed to relax the requirement of the evaluation of Hessian matrix [Agarwal et al., 2016, Carmon et al., 2016, 2017]. On the other hand, it seems difficult to ask the same property for first-order methods since they are blind to second-order information. Amazingly, it is shown that classical gradient descent method with a random initialization converges almost surely to a local minimum [Lee et al., 2016]. Moreover, with additional perturbed noise, gradient descent method or stochastic gradient descent (SGD) are guaranteed to escape saddle points [Ge et al., 2015, Jin et al., 2017].

Thus, there may be a hope to prove similar properties for our algorithm since Catalyst can be interpreted as an inexact accelerated proximal point algorithm. In fact, the experimental results in Section 5.6 do suggest our algorithm may be helpful to escape bad stationary points. Indeed, in the experiments for neural networks, we observe that when applying Catalyst, the subgradient norm may increase at some point then decrease, while the function value keeps decreasing. A more systematic study is required to confirm such a observation, which we leave as a potential direction of future work.

Appendix A

Proofs of simple results

A.1 Convergence of the gradient descent method

Theorem 10 (Convergence of gradient descent)**.** *If f is convex and L-smooth, then by letting $\eta_k = \frac{1}{L}$, the sequence $\{x_k\}_{k\in\mathbb{N}}$ generated by the gradient descent algorithm satisfies*

$$f(x_k) - f^* \leq \frac{L\|x_0 - x^*\|^2}{2k}, \quad \forall k \geq 1, \tag{A.1}$$

where f^* is the minimum of f and x^* is an optimal point with $f(x^*) = f^*$. Moreover, if f is μ-strongly convex, then

$$f(x_k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f^*)\,. \tag{A.2}$$

Proof. By the L-smoothness of the function f and the iterate update $x_{k+1} = x_k - \nabla f(x_k)/L$, we have

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\ &= f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2. \end{aligned}$$

This shows that the sequence $\{f(x_k)\}_{k\in\mathbb{N}}$ is decreasing. Moreover by convexity, we have

$$f(x_k) + \langle \nabla f(x_k), x^* - x_k\rangle \leq f^*.$$

Thus,

$$\begin{aligned} f(x_{k+1}) &\leq f^* + \langle \nabla f(x_k), x_k - x^*\rangle - \frac{1}{2L}\|\nabla f(x_k)\|^2 \\ &= f^* + \frac{L}{2}(\|x_k - x^*\|^2 - \|x_k - x^* - \frac{1}{L}\nabla f(x_k)\|^2) \\ &= f^* + \frac{L}{2}(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2). \end{aligned}$$

Summing it up from $i = 0$ to k yields

$$\sum_{i=0}^{k} f(x_{i+1}) - f^* \leq \frac{L}{2}\left(\|x_0 - x^*\|^2 - \|x_{k+1} - x^*\|^2\right).$$

As a result, since $f(x_i)$ is decreasing, we have

$$f(x_{k+1}) - f^* \leq \frac{L\|x_0 - x^*\|^2}{2(k+1)}.$$

When f is strongly convex,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2,$$

for any $x, y \in \mathbb{R}^d$. Taking both side the minimum according to y, we have

$$\min_{y\in\mathbb{R}^d} f(y) \geq \min_{y\in\mathbb{R}^d}\left\{f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2\right\},$$

which gives $f^* \geq f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2$. This is true for any $x \in \mathbb{R}^d$, thus

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2 \leq f(x_k) - \frac{\mu}{L}(f(x_k) - f^*).$$

Rearranging the terms gives

$$f(x_{k+1}) - f^* \leq \left(1 - \frac{\mu}{L}\right)(f(x_k) - f^*) \leq \cdots \leq \left(1 - \frac{\mu}{L}\right)^{k+1}(f(x_0) - f^*).$$

□

A.2 Quasi-Newton methods

Proposition 18. *The BFGS algorithm satisfies the first-order oracle defined in Assumption 1 if H_0 is initialized proportional to the identity matrix.*

Proof. We prove by recurrence that

$$x_k - x_0 \in Span\{\nabla f(x_0), \ldots, \nabla f(x_{k-1})\} \triangleq G_k.$$

When $k = 0$, this is true if H_0 is proportional to the identity matrix. Assume that this is true for all $i \leq k$. Then for $k + 1$, we have

$$x_{k+1} - x_0 = x_{k+1} - x_k + x_k - x_0 = -\eta_k H_k \nabla f(x_k) + \underbrace{x_k - x_0}_{\in G_k \subset G_{k+1}},$$

meaning that it suffices to show that $H_k \nabla f(x_k) \in G_{k+1}$. By the definition of the BFGS's update,

$$H_k = \underbrace{\left(I - \frac{s_{k-1} y_{k-1}^T}{\langle y_{k-1}, s_{k-1} \rangle}\right)}_{L_{k-1}} H_{k-1} \underbrace{\left(I - \frac{y_{k-1} s_{k-1}^T}{\langle y_{k-1}, s_{k-1} \rangle}\right)}_{R_{k-1}} + \frac{s_{k-1} s_{k-1}^T}{\langle y_{k-1}, s_{k-1} \rangle},$$

with $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$. The L_{k-1} and R_{k-1} represent the left and right matrix multiplied to H_{k-1} respectively. Unrolling the recursion gives

$$H_k = L_{k-1:0} H_0 R_{0:k-1} + \sum_{i=0}^{k-1} L_{k-1:i+1} \frac{s_i s_i^T}{y_i^T s_i} R_{i+1:k-1}, \tag{A.3}$$

where

$$L_{j:i} = \begin{cases} L_j L_{j-1} \cdots L_i & \text{if } j > i \\ I & \text{otherwise} \end{cases}, \quad R_{i:j} = \begin{cases} R_i R_{i+1} \cdots R_j & \text{if } j > i \\ I & \text{otherwise} \end{cases}. \tag{A.4}$$

We remark that for any vector z in $\mathbb{R}^d$,

$$\begin{aligned} L_{j:i} z &\in z + Span\{s_j, s_{j-1}, \cdots, s_i\}; \\ \text{and } R_{i:j} z &\in z + Span\{y_i, y_{i+1}, \cdots, y_j\}; \end{aligned}$$

This immediately shows that

$$H_k \nabla f(x_k) \in \nabla f(x_k) + Span\{s_0, \cdots, s_{k-1}, y_0, \cdots, y_{k-1}\} \subset G_{k+1},$$

where the last inclusion is true because all s_i and y_i are in G_{k+1}. □

A natural implication of relation (A.3) is the two loop recursion for BFGS update which can be extended to the limited memory variant:

Two loop recursion strategy

Input: $\nabla f(x_k)$ and two lists $\{s_0, \cdots s_{k-1}\}$, $\{y_0, \cdots, y_{k-1}\}$.

Initialize $z = \nabla f(x_k)$.

For $i = k-1$ to 0 do

$$\begin{aligned} \gamma_i &= \rho_i s_i^T z \quad \text{with} \quad \rho_i = \frac{1}{y_i^T s_i}, \\ z &= z - \gamma_i y_i. \end{aligned}$$

Update $z = H_0 z$.

For $i = 0$ to $k-1$ do

$$\begin{aligned} \lambda_i &= \rho_i y_i^T z \\ z &= z + s_i(\gamma_i - \lambda_i). \end{aligned}$$

Output the vector z which verifies $z = H_k \nabla g_k$.

Thus, instead of storing the dense matrix B_k or H_k, an alternative way is to store two list of vectors of s_i and y_i and use the two loop recursion strategy to build the matrix. This relax the requirement of the matrix storage and provide the possibility to limited memory variant.

Appendix B

Proof and technical results of Chapter 3

B.1 Useful lemmas

Lemma 16 (Simple lemma on quadratic functions). *Consider three vectors x, y and z in $\mathbb{R}^d$. Then, for all $\theta > 0$,*

$$\|x-y\|^2 \geq (1-\theta)\|x-z\|^2 + \left(1-\frac{1}{\theta}\right)\|z-y\|^2.$$

Proof.

$$\begin{aligned}
\|x-y\|^2 &= \|x-z+z-y\|^2 \\
&= \|x-z\|^2 + \|z-y\|^2 + 2\langle x-z, z-y\rangle \\
&= \|x-z\|^2 + \|z-y\|^2 + \|\sqrt{\theta}(x-z) + \frac{1}{\sqrt{\theta}}(z-y)\|^2 - \theta\|x-z\|^2 - \frac{1}{\theta}\|z-y\| \\
&\geq (1-\theta)\|x-z\|^2 + \left(1-\frac{1}{\theta}\right)\|z-y\|^2.
\end{aligned}$$

□

Lemma 17 (Simple lemma on non-negative sequences). *Consider a increasing sequence $(S_k)_{k\geq 0}$ and two non-negative sequences $(a_k)_{k\geq 0}$ and $(u_k)_{k\geq 0}$ such that for all k,*

$$u_k^2 \leq S_k + \sum_{i=1}^{k} a_i u_i. \tag{B.1}$$

Then,

$$S_k + \sum_{i=1}^{k} a_i u_i \leq \left(\sqrt{S_k} + \sum_{i=1}^{k} a_i\right)^2. \tag{B.2}$$

Proof. This lemma is identical to the Lemma A.10 in the original Catalyst paper [Lin et al., 2015], inspired by a lemma of Schmidt et al. [2011b] for controlling errors of inexact proximal gradient methods.

First, We give here an elementary proof for completeness based on induction. The relation (B.2) is obviously true for $k = 0$. Then, we assume it is true for $k-1$ and prove the relation for k. First, we remark that from (B.1),

$$\left(u_k - \frac{a_k}{2}\right)^2 \leq S_k + \sum_{i=1}^{k-1} a_i u_i + \frac{a_k^2}{4},$$

and then

$$u_k \leq \sqrt{S_k + \sum_{i=1}^{k-1} a_i u_i + \frac{a_k^2}{4}} + \frac{a_k}{2}.$$

We may now prove the relation (B.2) by induction,

$$\begin{aligned}
S_k + \sum_{i=1}^{k} a_i u_i &\leq S_k + \sum_{i=1}^{k-1} a_i u_i + a_k \left(\frac{a_k}{2} + \sqrt{S_k + \sum_{i=1}^{k-1} a_i u_i + \frac{a_k^2}{4}}\right) \\
&\leq S_k + \sum_{i=1}^{k-1} a_i u_i + a_k \left(a_k + \sqrt{S_k + \sum_{i=1}^{k-1} a_i u_i}\right) \\
&\leq \left(\sqrt{S_k + \sum_{i=1}^{k-1} a_i u_i} + a_k\right)^2 \\
&= \left(\sqrt{(S_k - S_{k-1}) + (S_{k-1} + \sum_{i=1}^{k-1} a_i u_i)} + a_k\right)^2 \\
&\leq \left(\sqrt{(S_k - S_{k-1}) + \left(\sqrt{S_{k-1}} + \sum_{i=1}^{k-1} a_i\right)^2} + a_k\right)^2 \qquad \text{(by induction)} \\
&\leq \left(\sqrt{S_k} + \sum_{i=1}^{k} a_i\right)^2.
\end{aligned}$$

The last inequality is obtained by developing the square $\left(\sqrt{S_{k-1} + \sum_{i=1}^{k-1} a_i}\right)^2$ and use the increasing assumption $S_{k-1} \leq S_k$. □

Lemma 18 (Growth of the sequence $(A_k)_{k\geq 0}$).
Let $(A_k)_{k\geq 0}$ be the sequence defined in (3.15) where $(\alpha_k)_{k\geq 0}$ is produced by (3.10) with $\alpha_0 = 1$ and $\mu = 0$. Then, we have the following bounds for all $k \geq 0$,

$$\frac{2}{(k+2)^2} \leq A_k \leq \frac{4}{(k+2)^2}.$$

Proof. The righthand side is directly obtained from Lemma 7 by noticing that $\gamma_0 = \kappa$ with the choice of α_0. Using the recurrence of α_k, we have for all $k \geq 1$,

$$\alpha_k^2 = (1-\alpha_k)\alpha_{k-1}^2 = \prod_{i=1}^{k}(1-\alpha_i)\alpha_0^2 = A_k \leq \frac{4}{(k+2)^2}.$$

Thus, $\alpha_k \leq \frac{2}{k+2}$ for all $k \geq 1$ (it is also true for $k = 0$). We now have all we need to conclude the lemma:

$$A_k = \prod_{i=1}^{k}(1-\alpha_i) \geq \prod_{i=1}^{k}\left(1 - \frac{2}{i+2}\right) = \frac{2}{(k+2)(k+1)} \geq \frac{2}{(k+2)^2}.$$

□

B.2 Proofs of auxiliary results

B.2.1 Proof of lemma 6

Proof. Let us introduce the quantity $h'(z) \triangleq \frac{1}{\eta}(z - [z]_\eta)$, which is often called "gradient mapping". Since $[z]_\eta$ is a proximal operator, we have

$$\frac{1}{\eta}\left((z - \eta\nabla h_0(z)) - [z]_\eta\right) \in \partial\psi([z]_\eta).$$

Therefore, since $h = h_0 + \psi$, we may define

$$u \triangleq \frac{1}{\eta}(z - [z]_\eta) - (\nabla h_0(z) - \nabla h_0([z]_\eta)) \in \partial h([z]_\eta),$$

and, by strong convexity,

$$\begin{aligned} h^* &\geq h([z]_\eta) + u^\top(p(x) - [z]_\eta) + \frac{\kappa}{2}\|p(x) - [z]_\eta\|^2 \\ &\geq h([z]_\eta) - \frac{1}{2\kappa}\|u\|^2. \end{aligned}$$

Moreover,

$$\begin{aligned} \|u\|^2 &= \left\|\frac{1}{\eta}(z - [z]_\eta)\right\|^2 - \frac{2}{\eta}\langle z - [z]_\eta, \nabla h_0(z) - \nabla h_0([z]_\eta)\rangle + \|\nabla h_0(z) - \nabla h_0([z]_\eta)\|^2 \\ &\leq \|h'(z)\|^2 - \|\nabla h_0(z) - \nabla h_0([z]_\eta)\|^2 \leq \|h'(z)\|^2 \leq (\kappa\varepsilon)^2, \end{aligned}$$

where the first inequality comes from the relation [Nesterov, 2004, Theorem 2.1.5]

$$\eta\|\nabla h_0(z) - \nabla h_0([z]_\eta)\|^2 \leq \langle \nabla h_0(z) - \nabla h_0([z]_\eta), z - [z]_\eta\rangle,$$

since h_0 is $(1/\eta)$-smooth. Thus,

$$h([z]_\eta) - h^* \leq \frac{1}{2\kappa}\|u\|^2 \leq \frac{1}{2\kappa}\|h'(z)\|^2 = \frac{1}{2\kappa\eta^2}\|z - [z]_\eta\|^2 \leq \varepsilon. \tag{B.3}$$

□

B.2.2 Proof of Proposition 2

Proof. We simply use Theorem 7 and specialize it to the choice of parameters. The initialization $\alpha_0 = \sqrt{q}$ leads to a particularly simple form of the algorithm, where $\alpha_k = \sqrt{q}$ for all $k \geq 0$. Therefore, the sequence $(A_k)_{k\geq 0}$ from Theorem 7 is also simple since we indeed have $A_k = (1-\sqrt{q})^k$. Then, we remark that $\gamma_0 = \mu(1-\sqrt{q})$ and thus, by strong convexity of f,

$$S_0 = (1-\sqrt{q})\left(f(x_0) - f^* + \frac{\mu}{2}\|x_0 - x^*\|^2\right) \leqslant 2(1-\sqrt{q})(f(x_0)-f^*).$$

Therefore,

$$\begin{aligned}
\sqrt{S_0} + 3\sum_{j=1}^{k}\sqrt{\frac{\varepsilon_j}{A_{j-1}}} &\leqslant \sqrt{2(1-\sqrt{q})(f(x_0)-f^*)} + 3\sum_{j=1}^{k}\sqrt{\frac{\varepsilon_j}{A_{j-1}}} \\
&= \sqrt{2(1-\sqrt{q})(f(x_0)-f^*)}\left[1 + \sum_{j=1}^{k}\underbrace{\left(\sqrt{\frac{1-\rho}{1-\sqrt{q}}}\right)^j}_{\eta}\right] \\
&= \sqrt{2(1-\sqrt{q})(f(x_0)-f^*)}\,\frac{\eta^{k+1}-1}{\eta-1} \\
&\leqslant \sqrt{2(1-\sqrt{q})(f(x_0)-f^*)}\,\frac{\eta^{k+1}}{\eta-1}.
\end{aligned}$$

Therefore, Theorem 7 combined with the previous inequality gives us

$$\begin{aligned}
f(x_k) - f^* &\leqslant 2A_{k-1}(1-\sqrt{q})(f(x_0)-f^*)\left(\frac{\eta^{k+1}}{\eta-1}\right)^2 \\
&= 2\left(\frac{\eta}{\eta-1}\right)^2(1-\rho)^k(f(x_0)-f^*) \\
&= 2\left(\frac{\sqrt{1-\rho}}{\sqrt{1-\rho}-\sqrt{1-\sqrt{q}}}\right)^2(1-\rho)^k(f(x_0)-f^*) \\
&= 2\left(\frac{1}{\sqrt{1-\rho}-\sqrt{1-\sqrt{q}}}\right)^2(1-\rho)^{k+1}(f(x_0)-f^*).
\end{aligned}$$

Since $\sqrt{1-x} + \frac{x}{2}$ is decreasing in $[0,1]$, we have $\sqrt{1-\rho} + \frac{\rho}{2} \geqslant \sqrt{1-\sqrt{q}} + \frac{\sqrt{q}}{2}$. Consequently,

$$f(x_k) - f^* \leqslant \frac{8}{(\sqrt{q}-\rho)^2}(1-\rho)^{k+1}(f(x_0)-f^*).$$

□

B.2.3 Proof of Proposition 3

Proof. The initialization $\alpha_0 = 1$ leads to $\gamma_0 = \kappa$ and $S_0 = \frac{\kappa}{2}\|x^* - x_0\|^2$. Then,

$$\begin{aligned}
&\sqrt{\frac{\gamma_0}{2}\|x_0 - x^*\|^2} + 3\sum_{j=1}^{k}\sqrt{\frac{\varepsilon_j}{A_{j-1}}} \\
&\qquad \leq \sqrt{\frac{\kappa}{2}\|x_0 - x^*\|^2} + 3\sum_{j=1}^{k}\sqrt{\frac{(j+1)^2\varepsilon_j}{2}} \quad \text{(from Lemma 18)} \\
&\qquad \leqslant \sqrt{\frac{\kappa}{2}\|x_0 - x^*\|^2} + \sqrt{f(x_0) - f^*}\left(\sum_{j=1}^{k}\frac{1}{(j+1)^{1+\gamma/2}}\right),
\end{aligned}$$

where the last inequality uses Lemma 18 to upper-bound the ratio ε_j/A_j. Moreover,

$$\sum_{j=1}^{k}\frac{1}{(j+1)^{1+\gamma/2}} \leqslant \sum_{j=2}^{\infty}\frac{1}{j^{1+\gamma/2}} \leqslant \int_1^{\infty}\frac{1}{x^{1+\gamma/2}}\,\mathrm{d}x = \frac{2}{\gamma}.$$

Then applying Theorem 7 yields

$$\begin{aligned}
f(x_k) - f^* &\leqslant A_{k-1}\left(\sqrt{\frac{\kappa}{2}\|x_0 - x^*\|^2} + \frac{2}{\gamma}\sqrt{f(x_0) - f^*}\right)^2 \\
&\leqslant \frac{8}{(k+1)^2}\left(\frac{\kappa}{2}\|x_0 - x^*\|^2 + \frac{4}{\gamma^2}(f(x_0) - f^*)\right).
\end{aligned}$$

The last inequality uses $(a+b)^2 \leqslant 2(a^2+b^2)$. □

B.2.4 Proof of Lemma 8

Proof. We abbreviate $\tau_{\mathcal{M}}$ by τ and $C = C_{\mathcal{M}}(h(z_0) - h^*)$ to simplify the notation. Set

$$T_0 = \frac{1}{\tau}\log\left(\frac{1}{1 - e^{-\tau}}\frac{C}{\varepsilon}\right).$$

For any $t \geq 0$, we have

$$\mathbb{E}[h(z_t) - h^*] \leqslant C(1-\tau)^t \leqslant C\,e^{-t\tau}.$$

By Markov's inequality,

$$\mathbb{P}[h(z_t) - h^* > \varepsilon] = \mathbb{P}[T(\varepsilon) > t] \leqslant \frac{\mathbb{E}[h(z_t) - h^*]}{\varepsilon} \leqslant \frac{C\,e^{-t\tau}}{\varepsilon}. \tag{B.4}$$

Together with the fact $\mathbb{P} \leqslant 1$ and $t \geq 0$. We have

$$\mathbb{P}[T(\varepsilon) \geqslant t+1] \leqslant \min\left\{\frac{C}{\varepsilon}e^{-t\tau}, 1\right\}.$$

Therefore,

$$\begin{aligned}\mathbb{E}[T(\varepsilon)] &= \sum_{t=1}^{\infty} \mathbb{P}[T(\varepsilon) \geqslant t] = \sum_{t=1}^{T_0} \mathbb{P}[T(\varepsilon) \geqslant t] + \sum_{t=T_0+1}^{\infty} \mathbb{P}[T(\varepsilon) \geqslant t] \\ &\leqslant T_0 + \sum_{t=T_0}^{\infty} \frac{C}{\varepsilon} e^{-t\tau} = T_0 + \frac{C}{\varepsilon} e^{-T_0\tau} \sum_{t=0}^{\infty} e^{-t\tau} \\ &= T_0 + \frac{C}{\varepsilon} \frac{e^{-\tau T_0}}{1 - e^{-\tau}} = T_0 + 1.\end{aligned}$$

As simple calculation shows that for any $\tau \in (0, 1)$, $\frac{\tau}{2} \leqslant 1 - e^{-\tau}$ and then

$$\mathbb{E}[T(\varepsilon)] \leqslant T_0 + 1 = \frac{1}{\tau} \log\left(\frac{1}{1 - e^{-\tau}} \frac{C}{\varepsilon}\right) + 1 \leqslant \frac{1}{\tau} \log\left(\frac{2C}{\tau\varepsilon}\right) + 1.$$

□

Appendix C

Short review regarding Quasi-Newton approximations

We have been presenting Quasi-Newton methods as a "natural" extension of Newton's method. But readers may wonder how the update of BFGS algorithm has been discovered. The expression of the update (4.2) is indeed not trivial at all. In order to provide a better understanding about Quasi-Newton methods, we perform in this chapter a literature review of them, aiming to give intuitions behind their constructions.

The whole story starts from the estimation of the Hessian matrix. For the purpose of reducing the computational complexity, we would like to approximate the Hessian matrix instead of evaluating it. More formally speaking, given a twice differentiable function f and a point x, we are looking for a reasonable approximation[1] B of the Hessian matrix $\nabla^2 f(x)$, based on gradient informations. A naive way is to apply the finite difference method, approximating each column by

$$B(:,i) = \left(\nabla f(x + he_i) - \nabla f(x)\right)/h, \quad \forall i \in [1,d] \tag{C.1}$$

where $h > 0$ and $(e_1, \ldots, e_d)$ is the canonical basis. This yields the discrete Newton's method which replaces the Hessian matrix in Newton's by the above approximation. However, in terms of computational complexity, discrete Newton's method requires d evaluations of gradients at each iteration, which is as expensive as evaluation of the Hessian matrix. Thus, in order to reduce the per-iteration cost, only a small amount of gradients are allowed to be evaluated, say one or two gradients, at each iteration. The main difficulty is that the Hessian matrix lives in the vector space of d^2 dimension but the informations we have is in the order of d. It then seems hopeless to provide a good estimation of the Hessian matrix, since the problem is largely underdetermined. The idea for going through it is to iteratively build the matrix B_{k+1} based on the past estimation B_k. If the Hessian of the function does not change too much, then B_k may still be a reasonable approximation of the current Hessian. This explains

[1] Remind that for historical reasons, we use B for denoting Hessian approximations and use H for denoting the approximations of the inverse Hessian matrix.

why an additional assumption is usually required for analyzing Quasi-Newton methods such as Lipschitz continuous Hessians.

Let us now introduce the secant equation, which is plays a crucial rule in the construction of Quasi-Newton mtehods. At $(k+1)$-th iteration, given the past iterate x_k and its Hessian approximation B_k, we evaluate the new iterate x_{k+1} through the Quasi-Newton update

$$x_{k+1} = x_k - B_k^{-1}\nabla f(x_k).$$

Then, the new Hessian approximation B_{k+1} at x_{k+1} is required to satisfy the secant equation,

$$B_{k+1}s_k = y_k, \tag{C.2}$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. The methods that we are presenting in this section all satisfy the secant equation (C.2), making it the core object to be understood.

Many explanations have been given to justify the requirement of the secant equation. One way is to introduce the local quadratic approximation

$$Q_{k+1}(x) = f(x_{k+1}) + \langle \nabla f(x_{k+1}), x - x_{k+1}\rangle + \frac{1}{2}(x - x_{k+1})^T B_{k+1}(x - x_{k+1}).$$

The secant equation is just asking the gradient of Q_{k+1} to match the gradient of f, at the previous iterate x_k. While this interpretation is easy to follow, it does not provide further information about how we can construct B_{k+1}. Since the secant equation is an underdetermined system which may admit an infinity amount of solutions, itself alone does not uniquely determine B_{k+1}.

A different way to interpret the secant equation is through the point of view of discretization. More precisely, by the definition of the Hessian matrix,

$$\lim_{h\to 0} \frac{\nabla f(x_{k+1}+h) - \nabla f(x_{k+1}) - \nabla^2 f(x_{k+1})h}{\|h\|} = 0.$$

The limit is valid along any direction h. The secant equation (C.2) is nothing else than a discretization of the above limit along the direction $h = x_k - x_{k+1} = -s_k$. The reason for choosing this direction is two-fold: first, x_k is the closest point to x_{k+1} among the past iterates, which makes the discretization error small; second, by letting $h = -s_k$, the gradient $\nabla f(x_{k+1} + h)$ becomes $\nabla f(x_k)$, which is already evaluated and does not require an additional gradient evaluation. Since the secant equation is the only information we have at iteration $k+1$, there is no justification to update B_{k+1} in other directions which are orthogonal to s_k. This leads to the condition

$$B_{k+1}z = B_k z, \quad \forall \langle z, s_k\rangle = 0. \tag{C.3}$$

Together with the secant equation (C.2), the matrix B_{k+1} is uniquely determined, giving the update of (Good) Broyden's method,

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{\|s_k\|^2}. \tag{C.4}$$

From the variational point of view, the constructed B_{k+1} is the closest matrix to B_k satisfying the secant equation [Dennis and Moré, 1977]:

Proposition 19. *Given B_k in $L(\mathbb{R}^d)$, y_k in $\mathbb{R}^d$ and a nonzero vector $s_k \in \mathbb{R}^d$, the matrix B_{k+1} defined in (C.4) is the unique solution of the problem*

$$\min_{B \in L(\mathbb{R}^d)} \{\|B - B_k\|_F \mid Bs_k = y_k\}, \tag{C.5}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

The interpretation based on the discretization provides us a more exhaustive point of view about the construction of Quasi-Newton methods. The key idea is to iteratively correct the Hessian estimation by adding new information regarding the current iterate. Broyden's update is the simplest one which does not guarantee the symmetry nor the positive definiteness of the matrix. This may be the reason why the corresponding theoretical result for Broyden's update is weaker than other well known Quasi-Newton methods. When the objective function is quadratic, Broyden's method is globally and superlinearly convergent to the exact minimum under mild modifications, see [Moré and Trangenstein, 1976].[2] Discarding the convergence analysis, let us show that we benefit a reduction in the computational complexity. Especially, we improve the $O(d^3)$ operations required for the matrix inversion to $O(d^2)$ operations, by applying the Sherman-Morrison formula [Sherman and Morrison, 1950].

Lemma 19. *Let $u, v \in \mathbb{R}^d$ and $A \in L(\mathbb{R}^d)$ is non singular. Then $A + uv^T$ is non singular if and only if $\sigma = 1 + \langle v, A^{-1}u\rangle \neq 0$. If $\sigma \neq 0$, then*

$$(A + uv^T)^{-1} = A^{-1} - \frac{1}{\sigma} A^{-1} uv^T A^{-1}. \tag{C.6}$$

As a consequence, one can implement Broyden's method by considering the update on the inverse of Hessian: set $H_k = B_k^{-1}$, then apply Lemma 19 to (C.4) yields

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k) s_k^T H_k}{s_k^T H_k y_k}, \tag{C.7}$$

under the condition $s_k^T H_k y_k \neq 0$. Then, the next iterate can be obtained by performing

$$x_{k+2} = x_{k+1} - H_{k+1} \nabla f(x_{k+1}).$$

The matrix inversion is no longer required which reduces the complexity to $O(d^2)$ due to the multiplication of matrix vector. Broyden's method is originally designed for solving nonlinear equations, in which case the underlying structure can be neither symmetric nor positive definite. However, in the unconstrained convex optimization setting, these properties are naturally

[2] Convergence results are beyond the focus of this part, we invite readers who are interested in that to go through the discussion in Section 1.3.

satisfied by the Hessian matrix. Thus, it is natural to maintain them in the construction of the metrics such that

$$B_k \text{ symmetric} \quad \text{implies} \quad B_{k+1} \text{ symmetric;} \tag{S}$$

$$\text{and} \quad B_k \text{ positive definite} \quad \text{implies} \quad B_{k+1} \text{ positive definite.} \tag{P}$$

Different update strategy are then proposed to guarantee symmetry or positive definiteness. Let us first investigate the rank one update formula, meaning that B_{k+1} is constructed by adding a rank one matrix to B_k. Remark that Broyden's method (C.4) is indeed a rank one update, but it does not guarantee the symmetry (S). More generally, any rank one update satisfying the secant equation can be written in the form

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) z_k^T}{\langle z_k, s_k \rangle}, \tag{C.8}$$

where z_k is any vector non-orthogonal to s_k. More precisely, the solutions of the secant equation forms an affine subspace $V = \{B \,|\, y_k = Bs_k\}$ and the update (C.8) is the projection of B_k on V along the rank one subspace $\{uz_k^T \,|\, u \in \mathbb{R}^d\}$ generated by z_k. The vector z_k represents the "direction" in which we project the metric. It is immediate to see that $z_k = y_k - B_k s_k$ provides the unique symmetric update

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{\langle y_k - B_k s_k, s_k \rangle}, \tag{C.9}$$

as long as B_k is symmetric and $\langle y_k - B_k s_k, s_k \rangle \neq 0$. This update is known as the *symmetric single rank* formula, called the SR1 method [Broyden, 1967]. One may use the Sherman-Morrison formula (19) to develop an update of the inverse Hessain matrix, see Table C.1. As its name indicates, the SR1 method ensures the metric B_k being symmetric, but unfortunately it does not guarantee the positive definiteness, (a counter example can be found in Example 11.2 in [Nocedal and Wright, 2006]). Moreover, the vectors s_k and $y_k - B_k s_k$ can be orthogonal, in which case the algorithm is not defined. Since the SR1 update is the unique rank one update conserving the symmetry, there is no hope to ensure both the symmetry and the positive definiteness by a rank one update. This is the reason why rank two update are investigated.

A generic routine for constructing a rank two update has been proposed by Powell [1970] for deriving extensions of Broyden's method. Given the past metric B, vectors y and s, fix a direction of projection z, the procedure iteratively constructs the following sequence $(C_j)_{j \in N}$ with $C_0 = B$ and

$$C_{2j+1} = C_{2j} + \frac{(y - C_{2j} s) z^T}{\langle z, s \rangle}, \tag{C.10}$$

$$C_{2j+2} = (C_{2j+1} + C_{2j+1}^T)/2. \tag{C.11}$$

Intuitively, C_{2j+1} is the projection from C_{2j} to the affine space defined by the solutions of the secant equation, along the "direction" z, and C_{2j+2} is the projection of C_{2j+1} on the space of symmetric matrices. This is nothing but an alternating projection procedure. It turns out that the sequence C_j converge to a limit C_∞ with a rank two update

$$C_\infty = B + \frac{(y - Bs)z^T + z(y - Bs)}{\langle z, s\rangle} - \frac{\langle y - Bs, s\rangle}{\langle z, s\rangle} zz^T. \tag{C.12}$$

In the variational point of view, this procedure converges to the solution of the following minimization problem:

Proposition 20. *Given $B \in L(\mathbb{R}^d)$ be symmetric, and let y, s and z be the vectors in $\mathbb{R}^d$ such that $\langle z, s\rangle > 0$. Let $W \in L(\mathbb{R}^d)$ be any nonsingular, symmetric matrix such that $Wz = s$. Then the matrix C_∞ defined in (C.12) is the unique solution of the problem*

$$\min_{\hat{B} \in L(\mathbb{R}^d)} \{\|\hat{B} - B\|_{F,W} \mid \hat{B} \text{ symmetric}, \ \hat{B}s = y\}, \tag{C.13}$$

where $\|\cdot\|_{F,W}$ denotes the weighted Frobenius norm defined by

$$\|A\|_{F,W} = \|W^{1/2} A W^{1/2}\|_F.$$

The update given by (C.12) is always symmetric and unintentionally it is a rank two update. Now it suffices to choose an appropriate direction z. By taking $z_k = s_k$, the underlying single rank update is Broyden's method, this gives the *Powell symmetric Broyden update*, called PSB update [Powell, 1970]:

$$B_{k+1}^{\text{PSB}} = B_k + \frac{r_k s_k^T + s_k r_k^T}{\|s_k\|^2} - \frac{\langle r_k, s_k\rangle s_k s_k^T}{\|s_k\|^4}, \tag{C.14}$$

where $r_k = y_k - B_k s_k$ is the residual vector. Unfortunately, the PSB update does not necessarily maintain the positive definiteness (P). This can be verified by checking the determinant of the matrix. Amazingly, a positive determinant is in fact a sufficient condition to guarantee the positive definiteness (P) of the update (C.12). The main reason is that the update can be rewritten as

$$B_{k+1} = B_k + aa^T - bb^T,$$

for some vectors $a, b \in \mathbb{R}^d$. Given that B_k is positive definite, B_{k+1} has at most one negative eigenvalue, meaning that the positive determinant is sufficient to ensure the positive definiteness. To make this reasoning rigorous, a result from perturbation theory is required [Wilkinson, 1965]:

Lemma 20. *Let $A \in L(\mathbb{R}^d)$ be symmetric with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d$. Let $u \in \mathbb{R}^d$ be a non zero vector and $B = A + \sigma uu^T$. Then if $\sigma \geq 0$, then the eigenvalues of B satisfies*

$$\lambda_1 \leq \lambda_1^B \leq \lambda_2 \leq \cdots \leq \lambda_d \leq \lambda_d^B;$$

otherwise if $\sigma < 0$, then the eigenvalues of B satisfies

$$\lambda_1^B \leq \lambda_1 \leq \lambda_2^B \leq \cdots \leq \lambda_d^B \leq \lambda_d.$$

From the expression (C.12), one can derive that

$$\det(B_{k+1}) = \frac{\det(B_k)}{\langle z_k, s_k\rangle^2}\left[\langle z_k, y_k\rangle_{H_k}^2 - \|z_k\|_{H_k}^2\|y_k\|_{H_k}^2 + \|z_k\|_{H_k}^2\langle y_k, s_k\rangle\right], \tag{C.15}$$

where $H_k = B_k^{-1}$ and $\langle u, v\rangle_{H_k} = u^T H_k v$. Thus the determinant is positive is equivalent to

$$\|z_k\|_{H_k}^2\langle y_k, s_k\rangle > \|z_k\|_{H_k}^2\|y_k\|_{H_k}^2 - \langle z_k, y_k\rangle_{H_k}^2. \tag{C.16}$$

For the PSB update, we have $z_k = s_k$ and the inequality does not always hold because it depends essentially on the spectrum of the metric $H_k = B_k^{-1}$. In order to make the update inherit the positive definiteness, a simple choice is to set $z_k = y_k$, in which case the righthand side vanishes. This gives the *Davidon-Fletcher-Powell* (DFP) update [Davidon, 1991, Fletcher and Powell, 1963],

$$\begin{aligned} B_{k+1}^{\mathrm{DFP}} &= B_k + \frac{(y_k - B_k s_k)y_k^T + y_k(y_k - B_k s_k)^T}{\langle y_k, s_k\rangle} - \frac{\langle y_k - B_k s_k, s_k\rangle y_k y_k^T}{\langle y_k, s_k\rangle^2} \\ &= \left(I - \frac{y_k s_k^T}{\langle y_k, s_k\rangle}\right) B_k \left(I - \frac{s_k y_k^T}{\langle y_k, s_k\rangle}\right) + \frac{y_k y_k^T}{\langle y_k, s_k\rangle}. \end{aligned} \tag{C.17}$$

Thus, B_{k+1}^{DFP} is positive definite if B_k is positive definite and $\langle y_k, s_k\rangle > 0$. Remind that $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and $s_k = x_{k+1} - x_k$, then $\langle y_k, s_k\rangle > 0$ holds whenever the function is strongly convex. Finally, to get rid of the matrix inversion, we apply the Sherman-Morrison formula twice, leading to the expression

$$H_{k+1}^{\mathrm{DFP}} = H_k + \frac{s_k s_k^T}{\langle s_k, y_k\rangle} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}. \tag{C.18}$$

So far, we have been focusing on approximating the Hessian matrix and apply the Sherman-Morrison formula to develop the inverse Hessian update. Indeed, it is also natural to directly approximate the inverse of the Hessian matrix H. In this case, the secant equation $B_{k+1}s_k = y_k$ transforms to the inverse secant equation, which is

$$H_{k+1}y_k = s_k. \tag{C.19}$$

We can construct H_{k+1} in the same way as we construct B_{k+1}. For instance, one can derive the (Bad) Broyden's method by following the reasoning of (Good) Broyden's method; the SR1 method remains the same. More importantly, by mimicking the DFP update, one can derive the *Broyden-Fletcher-Goldfarb-Shanno* update (BFGS) [Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970],

$$H_{k+1}^{\mathrm{BFGS}} = \left(I - \frac{s_k y_k^T}{\langle y_k, s_k\rangle}\right) H_k \left(I - \frac{y_k s_k^T}{\langle y_k, s_k\rangle}\right) + \frac{s_k s_k^T}{\langle y_k, s_k\rangle}. \tag{C.20}$$

The corresponding Hessian update is given by

$$B_{k+1}^{\text{BFGS}} = B_k + \frac{y_k y_k^T}{\langle y_k, s_k \rangle} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}. \tag{C.21}$$

Even though the underlying mechanism of BFGS is the same as the DFP update, the matrix H_{k+1}^{BFGS} is in fact different from the inverse of the DFP update H_{k+1}^{DFP}. One explanation is that the symmetrization operation $(A + A^T)/2$ does not commute with the inverse operation of matrices.

In this part, we have provided a review of several well-known Quasi-Newton methods. The construction starts from approximating the Hessian or inverse Hessian matrix, then desired properties such as symmetry and positive definiteness are carefully incorporated. The procedure that we reviewed provides a straightforward way for developing new variants of Quasi-Newton methods. In particular, we can replace the secant equation by

$$B_{k+1}h = \nabla f(x_{k+1} + h) - \nabla f(x_{k+1}),$$

where h is direction to be determined. The drawback of always choosing $h = -s_k$ in the secant equation is that s_k are usually quite "similar" for recent iterations, meaning that we may keep correcting our approximation in similar directions and ignoring the others. Ideally, as in the discrete Newton method, an Hessian estimation need to perform updates in all directions. Privileging one of them may not be a good choice since the other directions may be seriously wrong from the beginning. This effect does happen in practice that sometimes Quasi-Newton methods are stuck in a region for a long time. This may be an interesting direction for the future work. Finally, we list all the mentioned Quasi-Newton algorithm in the following table:

	Hessian estimation: B_{k+1}	Inverse Hessian estimation: H_{k+1}	(S)	(P)
Broyden	$\boldsymbol{B + \frac{(y - Bs)s^T}{\|s\|^2}}$	$H + \frac{(s - Hy)s^T H}{s^T Hy}$	✗	✗
SR1	$\boldsymbol{B + \frac{(y - Bs)(y - Bs)^T}{\langle y - Bs, s\rangle}}$	$\boldsymbol{H + \frac{(s - Hy)(s - Hy)^T}{\langle s - Hy, y\rangle}}$	✔	✗
PSB	$\boldsymbol{B + \frac{rs^T + sr^T}{\|s\|^2} - \frac{\langle r, s\rangle ss^T}{\|s\|^4}}$	$(*)$	✔	✗
DFP	$\boldsymbol{\left(I - \frac{ys^T}{\langle y, s\rangle}\right) B \left(I - \frac{sy^T}{\langle y, s\rangle}\right) + \frac{yy^T}{\langle y, s\rangle}}$	$H + \frac{ss^T}{\langle s, y\rangle} - \frac{Hyy^T H}{y^T Hy}$	✔	✔
BFGS	$B + \frac{yy^T}{\langle y, s\rangle} - \frac{Bss^T B}{s^T Bs}$	$\boldsymbol{\left(I - \frac{sy^T}{\langle y, s\rangle}\right) H \left(I - \frac{ys^T}{\langle y, s\rangle}\right) + \frac{ss^T}{\langle y, s\rangle}}$	✔	✔

Table C.1 – For simplicity, we abbreviate B_k, H_k, s_k, y_k by B, H, s, y, respectively. The vector r is the residual defined by $r = y - Bs$. The bold formulas are obtained directly form the construction and the others are obtained from applying the Sherman-Morrison formula (C.6). We omit the inverse Hessian update for PSB method since it is relatively complicate, an explicit formula can be found in equation (23) of [Powell, 1970].

Appendix D

Technical report: Catalyst Acceleration for Gradient-Based Non-Convex Optimization

Courtney Paquette
University of Washington

Hongzhou Lin
Inria

Dmitriy Drusvyatskiy
University of Washington

Julien Mairal
Inria

Zaid Harchaoui
University of Washington

D.1 Introduction

We consider optimization problems of the form

$$\min_{x\in\mathbb{R}^p} \Big\{ f(x) := f_0(x) + \psi(x) \Big\}, \quad \text{where } f_0(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)\,. \tag{D.1}$$

Here each function $f_i\colon \mathbb{R}^p \to \mathbb{R}$ is smooth, the regularization $\psi\colon \mathbb{R}^p \to \overline{\mathbb{R}}$ may be nonsmooth, and $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. By considering extended-real-valued functions, this composite setting also encompasses constrained minimization by letting ψ be the indicator function of the constraints on x.

Minimization of regularized empirical risk objectives of form (D.1) is central in machine learning. Whereas a significant amount of work has been devoted to this composite setting for convex problems, leading in particular to fast incremental algorithms [see, *e.g.*, Defazio et al., 2014a, Lan and Zhou, 2015, Mairal, 2015, Schmidt et al., 2017, Woodworth and Srebro, 2016,

Xiao and Zhang, 2014], the question of minimizing efficiently (D.1) when the functions f_i and ψ may be nonconvex is still largely open today.

Yet, nonconvex problems in machine learning are of high interest. For instance, the variable x may represent the parameters of a neural network, where each term $f_i(x)$ measures the fit between x and a data point indexed by i, or (D.1) may correspond to a nonconvex matrix factorization problem (see Section D.6). Besides, even when the data-fitting functions f_i are convex, it is also typical to consider nonconvex regularization functions ψ, for example for feature selection in signal processing [Hastie et al., 2015.]. In this work, we address two questions from nonconvex optimization:

1. How to apply a method for convex optimization to a nonconvex problem?

2. How to design an algorithm which does not need to know whether the objective function is convex while obtaining the optimal convergence guarantee if the function is convex?

Several pioneering works attempted to transfer ideas from the convex world to the nonconvex one, see, *e.g.*, [Ghadimi and Lan, 2016, Ghadimi et al., 2015]. Our paper has a similar goal and studies the extension of Nesterov's acceleration for convex problems [Nesterov, 1983] to nonconvex composite ones. Unfortunately, the concept of acceleration for nonconvex problems is unclear from a worst-case complexity point of view: gradient descent requires $O(\varepsilon^{-2})$ iterations to guarantee a gradient norm smaller than ε [Cartis et al., 2010, 2014]. Under a stronger assumption that the objective function is C^2-smooth, state-of-the-art methods [*e.g.*, Carmon et al., 2016] achieve a marginal gain with complexity $O(\varepsilon^{-7/4}\log(1/\varepsilon))$, and do not appear to generalize to composite or finite-sum settings. For this reason, our work fits within a broader stream of recent research on methods that *do not perform worse than gradient descent in the nonconvex case* (in terms of worst-case complexity), while *automatically accelerating for minimizing convex functions*. The hope when applying such methods to nonconvex problems is to see acceleration in practice, by heuristically exploiting convexity that is "hidden" in the objective (for instance, local convexity near the optimum, or convexity along the trajectory of iterates).

The main contribution of this part is a *generic* meta-algorithm, dubbed 4WD-Catalyst, which is able to use a *gradient-based* optimization method $\mathcal{M}$, originally designed for convex problems, and turn it into an accelerated scheme that also applies to nonconvex objective functions. The proposed 4WD-Catalyst can be seen as a **4**-**W**heel-**D**rive extension of Catalyst [Lin et al., 2015] to all optimization "terrains" (convex and nonconvex), while Catalyst was originally proposed for convex optimization. Specifically, without knowing whether the objective function is convex or not, our algorithm may take a method $\mathcal{M}$ designed for convex optimization problems with the same structure as (D.1), *e.g.*, SAGA [Defazio et al., 2014a], SVRG [Xiao and Zhang, 2014], and apply $\mathcal{M}$ to a sequence of sub-problems such that it asymptotically provides a stationary point of the nonconvex objective. Overall, the number of iterations of $\mathcal{M}$ to obtain

a gradient norm smaller than ε is $\tilde{O}(\varepsilon^{-2})$ in the worst case, while automatically reducing to $\tilde{O}(\varepsilon^{-2/3})$ if the function is convex.[1]

Related work. Inspired by Nesterov's acceleration method for convex optimization [Nesterov, 2004], the first accelerated method performing universally well for nonconvex and convex problems was introduced in [Ghadimi and Lan, 2016]. Specifically, this work addresses composite problems such as (D.1) with $n = 1$, and, provided the iterates are bounded, it performs no worse than gradient descent on nonconvex instances with complexity $O(\varepsilon^{-2})$ on the gradient norm. When the problem is convex, it accelerates with complexity $O(\varepsilon^{-2/3})$. Extensions to accelerated Gauss-Newton type methods were also recently developed in [Drusvyatskiy and Paquette, 2016]. In a follow-up work, the authors of [Ghadimi et al., 2015] proposed a new scheme that monotonically interlaces proximal gradient descent steps and Nesterov's extrapolation; thereby achieving similar guarantees as [Ghadimi and Lan, 2016] but without the need to assume the iterates to be bounded. Extensions when the gradient of ψ is only Hölder continuous can also be devised.

In [Li and Lin, 2015], a similar strategy is proposed, focusing instead on convergence guarantees under the so-called Kurdyka-Łojasiewicz inequality—a property corresponding to polynomial-like growth of the function, as shown by [Bolte et al., 2016]. Our scheme is in the same spirit as these previous papers, since it monotonically interlaces proximal-point steps (instead of proximal-gradient as in [Ghadimi et al., 2015]) and extrapolation/acceleration steps. A fundamental difference is that our method is generic and accommodates inexact computations, since we allow the subproblems to be approximately solved by any method we wish to accelerate.

By considering C^2-smooth nonconvex objective functions f with Lipschitz continuous gradient ∇f and Hessian $\nabla^2 f$, the authors of [Carmon et al., 2016] propose an algorithm with complexity $O(\varepsilon^{-7/4}\log(1/\varepsilon))$, based on iteratively solving convex subproblems closely related to the original problem. It is not clear if the complexity of their algorithm improves in the convex setting. Note also that the algorithm proposed in [Carmon et al., 2016] is inherently for C^2-smooth minimization and requires exact gradient evaluations. This implies that the scheme does not allow incorporating nonsmooth regularizers and can not exploit finite sum structure.

Finally, a stochastic method related to SVRG [Johnson and Zhang, 2013] for minimizing large sums while automatically adapting to the weak convexity constant of the objective function is proposed in [Allen-Zhu, 2017]. When the weak convexity constant is small (*i.e.*, the function is nearly convex), the proposed method enjoys an improved efficiency estimate. This algorithm, however, does not automatically accelerate for convex problems, in the sense that the overall rate is slower than $O(\varepsilon^{-3/2})$ in terms of target accuracy ε on the gradient norm.

[1] In this section, the notation $\tilde{O}$ only displays the polynomial dependency with respect to ε for the clarity of exposition.

Organization of the chapter. Section D.2 presents mathematical tools for non-convex and non-smooth analysis, which are used throughout the chapter. In Sections D.3 and D.4, we introduce the main algorithm and important extensions, respectively. Finally, we present experimental results on matrix factorization and training of neural networks in Section D.6.

D.2 Tools for nonconvex and nonsmooth optimization

Convergence results for nonsmooth optimization typically rely on the concept of subdifferential, which does not admit a unique definition in a nonconvex context [Borwein and Lewis, 2010]. In this chapter, we circumvent this issue by focusing on a broad class of nonconvex functions known as weakly convex or lower C^2 functions, for which all these constructions coincide. Weakly convex functions cover most of the interesting cases of interest in machine learning and resemble convex functions in many aspects. In this section, we formally introduce them and discuss their subdifferential properties.

Definition 7 (Weak convexity)**.** A function $f\colon \mathbb{R}^p \to \overline{\mathbb{R}}$ is $\rho-$*weakly convex* if for any points $x, y \in \mathbb{R}^p$ and $\lambda \in [0,1]$, the approximate secant inequality holds:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) + \rho\lambda(1-\lambda)\left\|x-y\right\|^2.$$

Notice that ρ-weak convexity with $\rho = 0$ is exactly the definition of a convex function. An elementary algebraic manipulation shows that f is ρ-weakly convex if and only if the function $x \mapsto f(x) + \frac{\rho}{2}\left\|x\right\|^2$ is convex. In particular, a C^1-smooth function f is ρ-weakly convex if the gradient ∇f is ρ-Lipschitz, while a C^2-smooth function f is ρ-weakly convex if and only if $\nabla^2 f(x) \succeq -\rho I$ for all x. This closely resembles an equivalent condition for C^2-smooth and μ-strongly convex functions, namely $\nabla^2 f(x) \succeq \mu I$ with $\mu > 0$.

Useful characterizations of ρ-weakly convex functions rely on differential properties. Since the functions we consider in the paper are nonsmooth, we use a generalized derivative construction. We mostly follow the standard monograph on the subject by Rockafellar and Wets [Rockafellar and Wets, 1998].

Definition 8 (Subdifferential)**.** Consider a function $f\colon \mathbb{R}^p \to \overline{\mathbb{R}}$ and a point x with $f(x)$ finite. The *subdifferential* of f at x is the set

$$\partial f(x) := \{v \in \mathbb{R}^p : f(y) \geq f(x) + v^T(y-x) + o(\|y-x\|) \quad \forall y \in \mathbb{R}^p\}.$$

Thus, a vector v lies in $\partial f(x)$ whenever the linear function $y \mapsto f(x) + v^T(y-x)$ is a lower-model of f, up to first-order around x. In particular, the subdifferential $\partial f(x)$ of a differentiable function f is the singleton $\{\nabla f(x)\}$, while for a convex function f it coincides with the subdifferential in the sense of convex analysis [see Rockafellar and Wets, 1998, Exercise 8.8]. It is useful to keep in mind that the sum rule, $\partial(f+g)(x) = \partial f(x) + \nabla g(x)$, holds for any differentiable function g.

We are interested in deriving complexity bounds on the number of iterations required by a method $\mathcal{M}$ to guarantee

$$\operatorname{dist}\Big(0, \partial f(x)\Big) \leq \varepsilon \ .$$

Recall when $\varepsilon = 0$, we are at a stationary point and satisfy first-order optimality conditions. In our convergence analysis, we will also use the following differential characterization of ρ-weakly convex functions, which generalize classical properties of convex functions. A proof follows directly from Theorem 12.17 of [Rockafellar and Wets, 1998] by taking into account that f is ρ-weakly convex if and only if $f + \frac{\rho}{2}\|\cdot\|^2$ is convex.

Theorem 11 (Differential characterization of ρ-weakly convex functions)**.**
For any lower-semicontinuous function $f\colon \mathbb{R}^p \to \overline{\mathbb{R}}$, the following properties are equivalent:

1. *f is ρ-weakly convex.*
2. *(subgradient inequality) The inequality*

$$f(y) \geq f(x) + v^T(y - x) - \frac{\rho}{2}\|y - x\|^2$$

holds for all $x, y \in \mathbb{R}^p$ and $v \in \partial f(x)$.

3. *(hypo-monotonicity) The inequality*

$$(v - w)^T(x - y) \geq -\rho\|x - y\|^2$$

holds for all $x, y \in \mathbb{R}^p$ and $v \in \partial f(x)$, $w \in \partial f(y)$.

Weakly convex functions have appeared in a wide variety of contexts, and under different names. Some notable examples are globally lower-C^2 [Rockafellar, 1982], prox-regular [Poliquin and Rockafellar, 1996], proximally smooth functions [Clarke et al., 1995], and those functions whose epigraph has positive reach [Federer, 1959].

D.3 The Basic 4WD-Catalyst algorithm for non-convex optimization

We now present a generic scheme (Algorithm 9) for applying a convex optimization method to minimize

$$\min_{x \in \mathbb{R}^p} \ f(x), \tag{D.2}$$

where f is only ρ-weakly convex. Our goal is to develop a unified framework that automatically accelerates in convex settings. Consequently, the scheme must be agnostic to the constant ρ.

D.3.1 Basic 4WD-Catalyst : a meta algorithm

At the center of our meta algorithm (Algorithm 9) are two sequences of subproblems, with better convexity properties, obtained by adding simple quadratics to f. The proposed approach extends the Catalyst acceleration of [Lin et al., 2015] and comes with a simplified convergence analysis. We next describe in detail each step of the scheme.

Two-step subproblems. The proposed acceleration scheme builds two main sequences of iterates $(\bar{x}_k)_k$ and $(\tilde{x}_k)_k$, obtaining from approximately solving two subproblems. These subproblems are simple quadratic perturbation of the original problem f having the form:

$$\min_x \left\{ f_\kappa(x; y) := f(x) + \frac{\kappa}{2} \|x - y\|^2 \right\}.$$

Here κ is a regularization parameter and y is called the *prox-center*. By adding the quadratic, we make the problem more "convex": when f is non convex, with a large enough κ, the subproblem will be convex; when f is convex, we improve the conditioning of the problem.

At the k-th iteration, given a previous iterate x_{k-1} and the extrapolation term v_{k-1}, we construct the two following subproblems.

1. **P**roximal point step. We first perform an inexact proximal point step with prox-center x_{k-1}:

$$\bar{x}_k \approx \arg\min_x f_\kappa(x; x_{k-1}) \quad \text{[Proximal-point step]}$$

2. **A**ccelerated proximal point step. Then we build the next prox-center y_k as the convex combination

$$y_k = \alpha_k v_{k-1} + (1 - \alpha_k) x_{k-1}. \tag{D.3}$$

Next we use y_k as a prox-center and update the next extrapolation term:

$$\begin{aligned} \tilde{x}_k &\approx \arg\min_x f_\kappa(x; y_k) && \text{[Accelerated proximal-point step]} \\ v_k &= x_{k-1} + \tfrac{1}{\alpha_k}(\tilde{x}_k - x_{k-1}) && \text{[Extrapolation]} \end{aligned} \tag{D.4}$$

where $\alpha_{k+1} \in (0, 1)$ is a sequence of coefficients satisfying $(1 - \alpha_{k+1})/\alpha_{k+1}^2 = 1/{\alpha_k}^2$. Essentially, the sequences $(\alpha_k)_k, (y_k)_k, (v_k)_k$ are built upon the extrapolation principles of [Nesterov, 2004].

Picking the best. At the end of iteration k, we have at hand two iterates, resp. $\bar{x}_k$ and $\tilde{x}_k$. Following [Ghadimi and Lan, 2016], we simply choose the best of the two in terms of their objective values, that is we choose x_k such that

$$f(x_k) \leq \min \left\{ f(\bar{x}_k), f(\tilde{x}_k) \right\}.$$

Algorithm 9 Basic 4WD-Catalyst

input Fix a point $x_0 \in \operatorname{dom} f$, real numbers $\kappa > 0$, and an optimization method $\mathcal{M}$.

initialization: $\alpha_1 \equiv 1$, $v_0 \equiv x_0$.

repeat for $k = 1, 2, \ldots$

1. Choose $\bar{x}_k$ using $\mathcal{M}$ such that

$$\bar{x}_k \approx \arg\min_x f_\kappa(x; x_{k-1}) \tag{D.5}$$

where $\operatorname{dist}\Big(0, \partial f_\kappa(\bar{x}_k; x_{k-1})\Big) < \kappa \|\bar{x}_k - x_{k-1}\|$ and $f_\kappa(\bar{x}_k; x_{k-1}) \leq f_\kappa(x_{k-1}; x_{k-1})$.

2. Set

$$y_k = \alpha_k v_{k-1} + (1 - \alpha_k) x_{k-1}. \tag{D.6}$$

3. Choose $\tilde{x}_k$ using $\mathcal{M}$ such that

$$\tilde{x}_k \approx \arg\min_x f_\kappa(x; y_k) \qquad \text{where} \qquad \operatorname{dist}\Big(0, \partial f_\kappa(\tilde{x}_k; y_k)\Big) < \frac{\kappa}{k+1} \|\tilde{x}_k - y_k\| \,. \tag{D.7}$$

4. Set

$$v_k = x_{k-1} + \frac{1}{\alpha_k}(\tilde{x}_k - x_{k-1}). \tag{D.8}$$

5. Pick $\alpha_{k+1} \in (0, 1)$ satisfying

$$\frac{1 - \alpha_{k+1}}{\alpha_{k+1}^2} = \frac{1}{\alpha_k^2}. \tag{D.9}$$

6. Choose x_k to be any point satisfying

$$f(x_k) \leq \min\{f(\bar{x}_k), f(\tilde{x}_k)\}. \tag{D.10}$$

until the stopping criterion $\operatorname{dist}\Big(0, \partial f(\bar{x}_k)\Big) < \varepsilon$

The proposed scheme blends the two steps in a synergistic way, allowing us to recover the near-optimal rates of convergence in both worlds: convex and non-convex. Intuitively, when $\bar{x}_k$ is chosen, it means that Nesterov's extrapolation step "fails" to accelerate convergence.

Stopping criterion for the subproblems. In order to derive complexity bounds, it is important to properly define the stopping criterion for the proximal subproblems. When the subproblem is convex, a functional gap like $f_\kappa(z; x) - \inf_z f_\kappa(z; x)$ may be used as a control of the inexactness, as in [Lin et al., 2015]. Without convexity, this criterion can not be used since

such quantities can not be easily bounded. In particular, first order methods seek points whose subgradient is small. Since small subgradients do not necessarily imply small function values in a non-convex setting, first order methods only test is for small subgradients. In contrast, in the convex setting, small subgradients imply small function values; thus a first order method in the convex setting can "test" for small function values. Hence, we can not direct application of catalyst [Lin et al., 2015] which uses the functional gap as a stopping criteria. Because we are working in the nonconvex setting, we include a stationarity stopping criteria.

We propose the following two types of stopping criteria:

1. Descent condition: $f_\kappa(z; y) \leq f_\kappa(y; y)$;
2. Adaptive stationary condition: $\operatorname{dist}\Big(0, \partial f_\kappa(z; y)\Big) < \kappa \|z - y\|$.

Without the descent condition, the stationarity condition is insufficient for defining a good stopping criterion because of the existence of local maxima in nonconvex problems. In the nonconvex setting, local maxima and local minima satisfy the stationarity condition. The descent condition ensures the iterates generated by the algorithm always decrease the value of objective function f; thus ensuring we move away from local maxima. The second criterion, adaptive stationary condition, provides a flexible relative tolerance on termination of algorithm used for solving the subproblems; a detailed analysis is forthcoming.

In Basic 4WD-Catalyst , we use both the stationary condition and the descent condition as a stopping criteria to produce the point $\bar{x}$:

$$\operatorname{dist}\Big(0, \partial f_\kappa(\bar{x}_k; x_{k-1})\Big) < \kappa \|\bar{x}_k - x_{k-1}\| \text{ and } f_\kappa(\bar{x}_k; x_{k-1}) \leq f_\kappa(x_{k-1}; x_{k-1}). \tag{D.11}$$

For the point $\tilde{x}$, our "acceleration" point, we use a modified stationary condition:

$$\operatorname{dist}\Big(0, \partial f_\kappa(\tilde{x}_k; y_k)\Big) < \frac{\kappa}{k+1} \|\tilde{x}_k - y_k\|. \tag{D.12}$$

The $k+1$ factor guarantees Basic 4WD-Catalyst accelerates for the convex setting. To be precise, Equation D.27 in the proofs of Theorem 13 and Theorem 13 uses the $k+1$ to ensure a produce converges. Note, we do not need the descent condition for $\tilde{x}$, as the functional decrease in $\bar{x}$ is enough to ensure the sequence $\{f(x_k)\}_{k\geq 1}$ is monotonically decreasing.

D.3.2 Convergence analysis.

We present here the theoretical properties of Algorithm 9. In this first stage, we do not take into account the complexity of solving the subproblems (D.5) and (D.7). For the next two theorems, we assume that the stopping criteria for the proximal subproblems are satisfied in each iteration of Algorithm 9.

Theorem 12 (Outer-loop complexity for Basic 4WD-Catalyst; non-convex case)**.** *For any $\kappa > 0$ and $N \geq 1$, the iterates generated by Algorithm 9 satisfy*

$$\min_{j=1,\ldots,N} \operatorname{dist}^2\Big(0, \partial f(\bar{x}_j)\Big) \leq \frac{8\kappa}{N}(f(x_0) - f^*).$$

It is important to notice that this convergence result is valid for any κ and does not require it to be larger than the weak convexity parameter. As long as the stopping criteria for the proximal subproblems are satisfied, the quantities $\operatorname{dist}(0, \partial f(\bar{x}_j))$ tend to zero. The proof is inspired by that of inexact proximal algorithms [Bertsekas, 2015, Güler, 1991, Lin et al., 2015] and appears in the appendix (see Appendix D.8).

If the function f turns out to be convex, the scheme achieves a faster convergence rate both in function values and in stationarity:

Theorem 13 (Outer-loop complexity, convex case)**.** *If the function f is convex, then for any $\kappa > 0$ and $N \geq 1$, the iterates generated by Algorithm 9 satisfy*

$$f(x_N) - f(x^*) \leq \frac{16\kappa}{(N+1)^2} \|x^* - x_0\|^2 \tag{D.13}$$

and

$$\min_{j=1,\dots,2N} \operatorname{dist}^2\big(0, \partial f(\bar{x}_j)\big) \leq \frac{32\kappa^2}{N(N+1)^2} \|x^* - x_0\|^2$$

where x^* is any minimizer of the function f.

This theorem establishes a rate of $O(N^{-2})$ for suboptimality in function value and convergence in $O(N^{-3/2})$ for the minimal norm of subgradients. The first rate is optimal in terms of information-based complexity for the minimization of a convex composite function [Nesterov, 2004, 2013]. The second can be improved to $O(N^{-2}\log(N))$ through a regularization technique, if one knew in advance that the function is convex and had an estimate on the distance of the initial point to an optimal solution [Nesterov, 2012a]. The proof appears in the appendix; see Appendix D.9.1.

Towards an automatically adaptive algorithm. So far, our analysis has not taken into account the cost of obtaining the iterates $\bar{x}_j$ and $\tilde{x}_j$ by the algorithm $\mathcal{M}$. We emphasize again that the two results above do not require any assumption on κ, which leaves us a degree of freedom. In order to develop the global complexity, we need to evaluate the total number of iterations performed by $\mathcal{M}$ throughout the process. Clearly, this complexity heavily depends on the choice of κ, since it controls the magnitude of regularization we add to improve the convexity of the subproblem. This is the point where a careful analysis is needed, because our algorithm must adapt to ρ without knowing it in advance. The next section is entirely dedicated to this issue. In particular, we will explain how to automatically adapt the parameter κ (Algorithm 10).

D.4 The 4WD-Catalyst algorithm

In this section, we work towards understanding the global efficiency of Algorithm 9, which automatically adapts to the weak convexity parameter. For this, we must take into account the

cost of approximately solving the proximal subproblems to the desired stopping criteria. First, we require some natural conditions on the optimization method $\mathcal{M}$.

Linear convergence of $\mathcal{M}$: When κ is sufficiently large, the subproblems become strongly convex; thus easy to solve. As such, we assume that $\mathcal{M}$ has a linear convergence rate, namely for any $\kappa > \rho$, there exist $A_\kappa \geq 0$ and $\tau_\kappa \in (0, 1)$ so that the following hold:

1. For any prox-center $y \in \mathbb{R}^p$ and initial $z_0 \in \mathbb{R}^p$ the iterates $\{z_t\}_{t \geq 1}$ generated by $\mathcal{M}$ on the problem $\min_z f_\kappa(z; y)$ satisfy

$$\operatorname{dist}^2(0, \partial f_\kappa(z_t; y)) \leq A_\kappa (1 - \tau_\kappa)^t (f_\kappa(z_0; y) - f_\kappa^*(y)) \tag{D.14}$$

   where $f_\kappa(y)^* := \inf_z f_\kappa(z; y)$. If the method $\mathcal{M}$ is randomized, we allow this inequality to take an expectation

$$\mathbb{E}\left[\operatorname{dist}^2\Big(0, \partial f_\kappa(z_t; y)\Big)\right] \leq A_\kappa (1 - \tau_\kappa)^t \left(f_\kappa(z_0; y) - f_\kappa^*(y)\right).$$

2. The rates τ_κ and the constants A_κ are increasing in κ.

Remark 2. The linear convergence we assume here for $\mathcal{M}$ differs from the one considered by [Lin et al., 2015], which was given in terms of function values. However, if the problem is a composite one, both points of view are near-equivalent, as discussed in Section D.7 and the precise relationship is given in Appendix D.9.

We chose the norm of the subgradient as our measurement because the complexity analysis is easier. This allow us to bound the computational complexity to achieve an ε approximate stationary point.

Lemma 21. *Given a strongly convex problem $f_\kappa(\cdot; y)$ and a linear convergent method $\mathcal{M}$ generating $\{z_t\}_{t \geq 0}$. Given a target accuracy ε, denote $T(\varepsilon) = \inf\{t \geq 1, \operatorname{dist}\Big(0, \partial f_\kappa(z_t; y)\Big) \leq \varepsilon\}$.*

1. *If $\mathcal{M}$ is deterministic, then $T(\varepsilon) \leq \frac{1}{\tau_\kappa} \log\left(\frac{A_\kappa (f_\kappa(z_0; y) - f_\kappa^*(y))}{\varepsilon^2}\right)$.*

2. *If $\mathcal{M}$ is randomized, then $\mathbb{E}[T(\varepsilon)] \leq \frac{1}{\tau_\kappa} \log\left(\frac{A_\kappa (f_\kappa(z_0; y) - f_\kappa^*(y))}{\tau_\kappa \varepsilon^2}\right)$. [Lemma C.1 of [Lin et al., 2015]]*

As we can see, we lose a factor in the log term by passing from deterministic to randomized algorithms. For the sake of simplicity, we perform our analysis only for deterministic algorithms and we use τ_κ to denote the linear rate for f_κ.

Bounding the required iterations when $\kappa > \rho$. When $\kappa > \rho$ and τ_κ is explicitly given as a function of κ, the number of inner calls to $\mathcal{M}$ is easy to compute. First, we define the notation

$$f_0(x; y) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \frac{\kappa}{2}\|x - y\|^2 \tag{D.15}$$

$$y^0 = \text{prox}_{1/(\kappa+L)f_0}\left(y - \frac{1}{\kappa + L}\nabla f_0(y; y)\right). \tag{D.16}$$

and note that $f_\kappa(x; y) = f_0(x; y) + \psi(x)$. The number of inner calls is given by the following formula:

Theorem 14. *Suppose $\kappa > \rho$. Define the initialization point z_0 by*

1. *if $f_\kappa(\cdot; y)$ is smooth, set $z_0 = y$;*

2. *if $f_\kappa(\cdot; y)$ is a composite form, $f_\kappa(\cdot; y) = f_0(\cdot; y) + \psi(\cdot)$ with the function f_0 having $(L+\kappa)$-Lipschitz continuous gradient, set $z_0 = prox_{\eta\psi}(y - \eta\nabla f_0(y))$ with $\eta \leq \frac{1}{L+\kappa}$.*

If $\mathcal{M}$ is initialized at the point z_0, then

1. *the output z_T satisfies $f_\kappa(z_T; y) \leq f_\kappa(z_0; y)$ (descent condition) and $dist(0, \partial f_\kappa(z_T; y)) \leq \kappa\|z_T - y\|$ (adaptive stationary condition) in at most T_κ iterations where*

$$T_\kappa = \frac{1}{\tau_\kappa}\log\left(\frac{32A_\kappa(L+\kappa)}{(\kappa-\rho)^2}\right);$$

2. *the output z_S satisfies $dist(0, \partial f_\kappa(z_S; y)) \leq \frac{\kappa}{k+1}\|z_S - y\|$ (modified adaptive stationary condition) in at most $S_\kappa \log(k+1)$ iterations where*

$$S_\kappa \log(k+1) = \frac{1}{\tau_\kappa}\log\left(\frac{32A_\kappa(L+\kappa)(k+1)^2}{(\kappa-\rho)^2}\right).$$

The proof is technical so we will leave it to the end of the supplementary materials (see Section D.7). When ρ is known, such that κ can be chosen large enough to guarantee linear convergene of $\mathcal{M}$, the global complexity for Algorithm 9 is immediate. Herein lies a problem– the smoothing parameter κ drives the outer complexity (see Theorem 12 and Theorem 13). By choosing κ too large, we may fail to obtain the desired optimal complexity guarantees; choosing κ too small the method $\mathcal{M}$ converges at suboptimal rates.

Algorithm 10 4WD-Catalyst

input Fix a point $x_0 \in \operatorname{dom} f$, real numbers $\kappa_0, \kappa_{\text{cvx}} > 0$ and $T, S > 0$, and an opt. method $\mathcal{M}$.

initialization: $\alpha_1 = 1$, $v_0 = x_0$.

repeat for $k = 1, 2, \ldots$

1. Compute
$$(\bar{x}_k, \kappa_k) = \text{Auto-adapt } (x_{k-1}, \kappa_{k-1}, T).$$
2. Compute $y_k = \alpha_k v_{k-1} + (1 - \alpha_k) x_{k-1}$ and apply $S \log(k+1)$ iterations of $\mathcal{M}$ to find
$$\tilde{x}_k \approx \arg\min_{x \in \mathbb{R}^p} f_{\kappa_{\text{cvx}}}(x, y_k). \tag{D.17}$$
3. Update v_k and α_{k+1} by
$$v_k = x_{k-1} + \tfrac{1}{\alpha_k}(\tilde{x}_k - x_{k-1}) \quad \text{and} \quad \alpha_{k+1} = \frac{\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2}{2}.$$
4. Choose x_k to be any point satisfying $f(x_k) = \min\{f(\bar{x}_k), f(\tilde{x}_k)\}$.

until the stopping criterion $\operatorname{dist}\big(0, \partial f(\bar{x}_k)\big) < \varepsilon$

Adaptive versions of Basic 4WD-Catalyst. To solve the previous theoretical and practical challenges, we introduce an adaptive variant, called 4WD-Catalyst, of Basic 4WD-Catalyst. The adaptive algorithm is presented in Algorithm 10. There are two main modifications to Algorithm 9.

The *first idea* consists of using a pre-defined number of iterations for solving (D.7), corresponding to the maximal number of iterations of $\mathcal{M}$ that we are willing to tolerate for evaluating $\tilde{x}_k$. The main motivation is three-fold: (i) the complexity of such a strategy is known in advance; (ii) in the convex case, we can choose a small κ, denoted by κ_{cvx}, that ensures we obtain the optimal convergence guarantee; (iii) for nonconvex problems, we remark from the proof of Theorem 12 that the convergence guarantees are obtained regardless of the quality of $\tilde{x}_k$ for solving (D.7). In other words, the conclusions of Theorem 12 are valid even when the iterates $\tilde{x}_k$ do not satisfy $\|\partial f_\kappa(\tilde{x}_k)\| < \frac{\kappa}{k+1}\|\tilde{x}_k - y_k\|$ for non-convex functions.This allows us to choose the same strategy for computing $\tilde{x}_k$ in the convex and non-convex case—that is, we use the pre-defined number of iterations of $\mathcal{M}$ and a "convex" smoothing parameter κ_{cvx}.

The *second idea* lies in an adaptive procedure, Auto-adapt; see Algorithm 11. This procedure uses a pre-defined number of iterations of $\mathcal{M}$ to compute the iterates $\bar{x}_k$ while automatically choosing the smoothing parameter κ. We show that as soon as the number of iterations allocated to $\mathcal{M}$ for computing $\bar{x}_k$ is large enough, acceleration is preserved in the convex case, while

keeping the conclusions of Theorem 12 valid. Under our assumptions on $\mathcal{M}$ (see Section D.4), we show that the procedure Auto-adapt$(x, \kappa, \varepsilon, T)$ indeed terminates regardless of the number of iterations T chosen with the assumption that $\tau_\kappa \to 1$ as $\kappa \to \infty$ (see Lemma 24). If the later is not the case, we can still choose T sufficiently large so that Auto-adapt$(x, \kappa, \varepsilon, T)$ terminates (See Proposition 21). When a formula for τ_κ is explicitly given as a function of κ, the number of calls made by Auto-adapt$(x, \kappa, \varepsilon, T)$ to $\mathcal{M}$ is easy to compute.

Algorithm 11 Auto-adapt (x, κ, T)

input $x \in \mathbb{R}^p$, method $\mathcal{M}$, $\kappa > 0$, number of iterations T.

Repeat Compute

$$z_T \approx \arg\min_{z \in \mathbb{R}^p} f_\kappa(z; x).$$

by running T iterations of $\mathcal{M}$ initializing from $z_0 = x$ or $z_0 = \mathrm{prox}_{\eta\psi}(x - \eta \nabla f_0(x))$.
If $f_\kappa(z_T; x) > f_\kappa(x; x)$ or $\mathrm{dist}(\partial f_\kappa(z_T; x), 0) > \kappa \|z_T - x\|$,
then go to repeat with $\kappa \to 2\kappa$.
else go to output.

output (z_T, κ).

New stopping criteria for the subproblems Recall, equations (D.11) and (D.12) rely on a fixed parameter $\kappa > \rho$. To derive the global complexity results for Basic 4WD-Catalyst that match optimal convergence guarantees, we make a distinction between the κ in the convex setting versus the nonconvex setting. As the weak convexity constant is unknown and κ depends on k, we introduce a parameter $\kappa_0 > 0$ as an initial guess for ρ. Algorithm 11 produces a sequence of κ_k. In replacement of (D.11), our new stopping criteria to produce the point $\bar{x}_k$ is

$$\mathrm{dist}\Big(0, \partial f_{\kappa_k}(\bar{x}_k; x_{k-1})\Big) < \kappa_k \|\bar{x}_k - x_k\| \text{ and } f_{\kappa_k}(\bar{x}_k; x_{k-1}) \leq f_{\kappa_k}(x_{k-1}; x_{k-1}). \tag{D.18}$$

To obtain the optimal convergence guarantee in the convex setting, we introduce the parameter κ_{cvx}, the choice of which is same as the smoothing parameter in [Lin et al., 2015] and depends on the method $\mathcal{M}$. In replacement of (D.12), the new stopping criteria for $\tilde{x}_k$ is

$$\mathrm{dist}\Big(0, \partial f_{\kappa_{\mathrm{cvx}}}(\tilde{x}_k; y_k)\Big) < \frac{\kappa_{\mathrm{cvx}}}{k+1} \|\tilde{x}_k - y_k\|. \tag{D.19}$$

Convergence analysis. Let us next postulate that T and S are chosen large enough to guarantee that $\bar{x}_k$ and $\tilde{x}_k$ satisfy conditions (D.18) and (D.19) for the corresponding subproblems, and see how the outer algorithm complexity resembles the guarantees of Theorem 12 and Theorem 13. The main technical difference is that κ changes at each iteration k and the new stopping criterions (D.18) and (D.19), namely keeping track of the effects of κ_0 and κ_{cvx} on the proof.

Theorem 15 (Outer-loop complexity, 4WD-Catalyst)**.** *Fix real constants $\kappa_0, \kappa_{\text{cvx}} > 0$, and $x_0 \in dom\ f$. Set $\kappa_{\max} := \max_{k\geq 1} \kappa_k$. Suppose that the number of iterations T is such that $\bar{x}_k$ satisfies (D.18). Define $f^* := \lim_{k\to\infty} f(x_k)$. Then for any $N \geq 1$, the iterates generated by Algorithm 10 satisfy,*

$$\min_{j=1,\dots,N} \operatorname{dist}^2\Big(0, \partial f(\bar{x}_j)\Big) \leq \frac{8\kappa_{\max}}{N}(f(x_0) - f^*).$$

If in addition the function f is convex and S_k is chosen so that $\tilde{x}_k$ satisfies (D.19), then

$$\min_{j=1,\dots,2N} \operatorname{dist}^2\Big(0, \partial f(\bar{x}_j)\Big) \leq \frac{32\kappa_{\max}\kappa_{\text{cvx}}}{N(N+1)^2}\|x^* - x_0\|^2$$

and

$$f(x_N) - f(x^*) \leq \frac{16\kappa_{\text{cvx}}}{(N+1)^2}\|x^* - x_0\|^2 \tag{D.20}$$

where x^* is any minimizer of the function f.

Inner-loop Complexity In light of Theorem 15, we must now understand how to choose T and S as small as possible, while guaranteeing that $\bar{x}_k$ and $\tilde{x}_k$ satisfy (D.18) and (D.19) hold for each k. The quantities T and S must be set beforehand without knowing the true value of the weak convexity constant ρ. Using Theorem 14, we assert the following choices for T and S.

Theorem 16 (Inner complexity for 4WD-Catalyst : determining the values T and S)**.** *Suppose the stopping criteria are (D.18) and (D.19) as in in Theorem 15 and we choose T and S in Algorithm 10 to be the smallest numbers satisfying*

$$T \geq \frac{1}{\tau_L}\log\left(\frac{160A_{4L}}{L}\right)$$

and

$$S\log(k+1) \geq \frac{1}{\tau_{\kappa_{\text{cvx}}}}\log\left(\frac{32A_{\kappa_{\text{cvx}}}(\kappa_{\text{cvx}}+L)(k+1)^2}{\kappa_{\text{cvx}}^2}\right)$$

for all k. In particular,

$$T = O\left(\frac{1}{\tau_L}\log\left(A_{4L}, L\right)\right),$$
$$S = O\left(\frac{1}{\tau_{\kappa_{\text{cvx}}}}\log(A_{\kappa_{\text{cvx}}}, L, \kappa_{\text{cvx}})\right).$$

Then $\kappa_{\max} \leq 4L$ and the following hold for any index $k \geq 1$:

1. *Generating $\bar{x}_k$ in Algorithm 10 starting at x_{k-1}^0 requires at most $\tilde{O}\left(\tau_L^{-1}\right)$ iterations of $\mathcal{M}$;*

2. Generating $\tilde{x}_k$ in Algorithm 10 starting at y_k^0 requires at most $\tilde{O}\left(\tau_{\kappa_{\text{cvx}}}^{-1}\right)$ iterations of $\mathcal{M}$.

where $\tilde{O}$ hides universal constants and logarithmic dependencies on k, L, κ_{cvx}, A_L, and $A_{\kappa_{\text{cvx}}}$.

Appendix D.10 is devoted to proving Theorem 16, but we outline below the general procedure and state the two main propositions (see Proposition 21 and Proposition 22).

We summarize the proof of Theorem 16 as followed:

1. When $\kappa > \rho + L$, we compute the number of iterations of $\mathcal{M}$ to produce a point satisfying (D.18). Such a point will become $\bar{x}_k$.
2. When the function f is convex, we compute the number of iterations of $\mathcal{M}$ to produce a point which satisfies the (D.19) condition. Such a point will become the point $\tilde{x}_k$.
3. We compute the smallest number of times we must double κ_0 until it becomes larger than $\rho + L$. Thus eventually the condition $4L \geq \kappa > \rho + L$ will occur.
4. We always set the number of iterations of $\mathcal{M}$ to produce $\bar{x}_k$ and $\tilde{x}_k$ as in Step 1 and Step 2, respectively, regardless of whether $f_\kappa(\cdot; x_k)$ is convex or f is convex.

The next proposition shows that Auto-adapt terminates with a suitable choice for $\bar{x}_k$ after T number of iterations.

Proposition 21 (Inner complexity for $\bar{x}_k$)**.** *Suppose $\rho + L < \kappa \leq 4L$. If the method $\mathcal{M}$ is initialized at a point $x^0 \in \mathbb{R}^p$, where x^0 is defined Theorem 14, and is applied to the problem*

$$\min_z f_\kappa(z; x) := f(z) + \frac{\kappa}{2} \|z - x\|^2$$

for at least the number of iterations

$$T \geq \frac{1}{\tau_L} \log\left(\frac{160 A_{4L}}{L}\right),$$

then the output z_M satisfies $f_\kappa(z_M; x) \leq f_\kappa(x; x)$ and $dist\Big(0, \partial f_\kappa(z_M; x)\Big) \leq \kappa \|z_M - x\|$.

Under the additional assumption that the function f is convex, we produce a point with (D.19) when the number of iterations, S, is chosen sufficiently large.

Proposition 22 (Inner-loop complexity for $\tilde{x}_k$)**.** *Fix a method $\mathcal{M}$ initialized at y_k^0, where y_k^0 is defined by either (D.16) in the composite setting or y_k when f is smooth, generating iterates $\{\tilde{z}_k\}_{k \geq 1}$ for minimizing $f_{\kappa_{\text{cvx}}}(\cdot; y_k)$ with linear convergence rates of the form (D.14). Suppose the function f is convex. If the number of iterations of $\mathcal{M}$ is*

$$S = O\left(\frac{1}{\tau_{\kappa_{\text{cvx}}}} \log(A_{\kappa_{\text{cvx}}}, L, \kappa_{\text{cvx}})\right)$$

such that

$$S\log(k+1) \geq \frac{1}{\tau_{\kappa_{\text{cvx}}}} \log\left(\frac{32A_{\kappa_{\text{cvx}}}(\kappa_{\text{cvx}}+L)(k+1)^2}{\kappa_{\text{cvx}}^2}\right), \tag{D.21}$$

then, the output $\tilde{z}_S = \tilde{x}_k$ *satisfies* $\|\partial f_{\kappa_{\text{cvx}}}(\tilde{z}_S)\| < \frac{\kappa_{\text{cvx}}}{k+1}\|\tilde{z}_{S_k} - y_k\|$ *for all* $k \geq 1$.

We can now derive global complexity bounds by combining Theorem 15 and Theorem 16, and a good choice for the constant κ_{cvx}.

Theorem 17 (Global complexity bounds for 4WD-Catalyst)**.** *Choose* T *and* S *as in* $\tilde{O}(\tau_L^{-1})$ *and* $\tilde{O}(\tau_{\kappa_{\text{cvx}}}^{-1})$ *(see Theorem 16 in appendix). We let* $\tilde{O}$ *hide universal constants and logarithmic dependencies in* A_L, $A_{\kappa_{\text{cvx}}}$, L, ε, κ_0, κ_{cvx}, *and* $\|x^* - x_0\|^2$. *Then the following are true.*

1. *Algorithm 10 will generate a point* x *satisfying* $\text{dist}\big(0, \partial f(x)\big) \leq \varepsilon$ *after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{\text{cvx}}}^{-1}\right) \cdot \frac{L(f(x_0) - f^*)}{\varepsilon^2}\right)$$

   iterations of the method $\mathcal{M}$.

2. *If* f *is convex, then Algorithm 10 will generate a point* x *satisfying* $\text{dist}\big(0, \partial f(x)\big) \leq \varepsilon$ *after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{\text{cvx}}}^{-1}\right) \cdot \frac{L^{1/3}\left(\kappa_{\text{cvx}}\|x^* - x_0\|^2\right)^{1/3}}{\varepsilon^{2/3}}\right)$$

   iterations of the method $\mathcal{M}$.

3. *If* f *is convex, then Algorithm 10 will generate a point* x *satisfying* $f(x) - f^* \leq \varepsilon$ *after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{\text{cvx}}}^{-1}\right) \cdot \frac{\sqrt{\kappa_{\text{cvx}}\|x^* - x_0\|^2}}{\sqrt{\varepsilon}}\right)$$

   iterations of the method $\mathcal{M}$.

D.5 Applications to Existing Algorithms

We show how to accelerate existing algorithms $\mathcal{M}$ and compare the convergence guaranties before and after 4WD-Catalyst. In particular, the algorithms considered are gradient descent and incremental methods, SAGA and SVRG. For all the algorithms considered, we state the convergence guaranties in terms of the *total number of iterations* (in expectation, if appropriate) to reach an accuracy of ε; in the convex setting, the accuracy is stated in terms of functional

error, $f(x) - \inf f < \varepsilon$ and in the nonconvex setting, the appropriate measure is stationarity, namely $\operatorname{dist}(0, \partial f(x)) < \varepsilon$. All the algorithms considered have formulations for the composite setting with analogous convergence rates. Table D.1 summarizes our conclusions for the algorithms considered.

Derivation of smoothing parameters. The smoothing parameter, κ_{cvx} drives the convergence rate of 4WD-Catalyst in the convex setting. To determine κ_{cvx}, we pretend $\rho = 0$ and compute the global complexity of our scheme. As such, we end up with the same complexity result as in [Lin et al., 2015]. Following their work, the rule of thumb is to maximize the ratio $\tau_\kappa / \sqrt{L + \kappa}$.

On the other hand, the choice of κ_0 is independent of $\mathcal{M}$; it is an initial lower estimate for the weak convexity constant, ρ. We provide a detailed derivation of all the variables for each of the considered algorithms in the appendix D.4.

Convergence results for existing algorithm. Table D.1 presents convergence rates for SAGA [Defazio et al., 2014a], (prox) SVRG [Xiao and Zhang, 2014], and gradient descent (FG). In the appendix, we provide a complete derivation of the rate for each of the algorithms.

	$\rho > 0$		$\rho = 0$	
	Original	4WD-Catalyst	Original	4WD-Catalyst
FG	$\mathcal{O}\left(n\frac{L}{\varepsilon^2}\right)$	$\tilde{O}\left(n\frac{L}{\varepsilon^2}\right)$	$O\left(n\frac{L}{\varepsilon}\right)$	$\tilde{O}\left(n\sqrt{\frac{L}{\varepsilon}}\right)$
SVRG [Xiao and Zhang, 2014]	not avail.	$\tilde{O}\left(n\frac{L}{\varepsilon^2}\right)$	not avail.	$\tilde{O}\left(\sqrt{n}\sqrt{\frac{L}{\varepsilon}}\right)$
SAGA [Defazio et al., 2014a]	not avail.	$\tilde{O}\left(n\frac{L}{\varepsilon^2}\right)$	$O\left(n\frac{L}{\varepsilon}\right)$	$\tilde{O}\left(\sqrt{n}\sqrt{\frac{L}{\varepsilon}}\right)$

Table D.1 – Comparison of rates of convergence, before and after the 4WD-Catalyst , resp. in the non-convex and convex cases. For the comparision, in the convex case, we only present the number of iterations to obtain a point x satisfying $f(x) - f^* < \varepsilon$. In the non-convex case, we show the number of iterations to obtain a point x satisfying $\operatorname{dist}(0, \partial f(x)) < \varepsilon$.

Full gradient method. A first illustration is the algorithm obtained when accelerating the regular "full" gradient (FG). Here, the optimal choice for κ_{cvx} is L. In the convex setting, we get an accelerated rate of $O(n\sqrt{L/\varepsilon}\log(1/\varepsilon))$ which agrees with Nesterov's accelerated variant (AFG) up to logarithmic factors. On the other hand, in the nonconvex setting, our approach achieves no worse rate than $O(nL/\varepsilon^2 \log(1/\varepsilon))$, which agrees with the standard gradient descent up to logarithmic factors. We note that under stronger assumptions, namely C^2-smoothness of the objective, the accelerated algorithm in [Carmon et al., 2017] achieves the same rate as (AFG) for the convex setting and $O(\varepsilon^{-7/4}\log(1/\varepsilon))$ for the nonconvex setting. Their approach, however, does not extend to composite setting nor to stochastic methods. Our marginal loss is the price we pay for considering a much larger class of functions.

Randomized incremental gradient. We now consider randomized incremental gradient methods such as SAGA [Defazio et al., 2014a] and (prox) SVRG [Xiao and Zhang, 2014]. Here, the optimal choice for κ_{cvx} is $O(L/n)$. Under the convex setting, we achieve an accelerated rate of $O(\sqrt{n}\sqrt{L/\varepsilon}\log(1/\varepsilon))$. A direct application of SVRG and SAGA have no convergence guarantees in the non-convex setting. With our approach, the resulting algorithm matches the guarantees for FG up to log factors.

Derivation of the variables for the existing algorithms For gradient descent, we have the following table of values

Variable	Description	Value
$1/\tau_L$	linear convergence with $\kappa = L$	2
κ_{cvx}	smooth parameter for convex setting	L
$1/\tau_{\kappa_{\text{cvx}}}$	linear convergence with $\kappa_{\text{cvx}} = L$	2
A_{4L}	Method $\mathcal{M}$ constant	$4L$

Using these values for gradient descent, the number of iterations in the inner loop are

$$T \geq 2\log(640)$$
$$S\log(k+1) \geq 2\log\left(64(k+1)^2\right)$$

The global complexity for gradient descent is

1. Algorithm 10 will generate a point x satisfying $\operatorname{dist}(0, \partial f(x)) \leq \varepsilon$ after at most

$$O\left[n\log\left(\frac{L}{\kappa_0}\right)\log\left(\frac{L^2(f(x_0)-f^*)^2}{\varepsilon^4}\right)\cdot\left(\frac{L(f(x_0)-f^*)}{\varepsilon^2}\right)\right]$$

   gradient computations.

2. If f is convex, then Algorithm 10 will generate a point x satisfying $\operatorname{dist}(0, \partial f(x)) \leq \varepsilon$ after at most

$$O\left[n\log\left(\frac{L}{\kappa_0}\right)\log\left(\frac{L^{4/3}\|x^*-x_0\|^{4/3}}{\varepsilon^{4/3}}\right)\cdot\left(\frac{L^{2/3}\|x^*-x_0\|^{2/3}}{\varepsilon^{2/3}}\right)\right]$$

   gradient computations.

3. If f is convex, then Algorithm 10 will generate a point x satisfying $f(x) - f^* \leq \varepsilon$ after at most

$$O\left[n \log\left(\frac{L}{\kappa_0}\right) \cdot \log\left(\frac{L\left\|x_0 - x^*\right\|^2}{\varepsilon}\right) \cdot \frac{\sqrt{L}\left\|x^* - x_0\right\|}{\sqrt{\varepsilon}}\right]$$

gradient computations.

For SVRG, we have the following table of values

Variable	Description	Value
$1/\tau_L$	linear convergence with $\kappa = L$	$n+2$
κ_{cvx}	smooth parameter for convex setting	L/n
$1/\tau_{\kappa_{\text{cvx}}}$	linear convergence with $\kappa_{\text{cvx}} = L/n$	$2n+1$
A_{4L}	Method $\mathcal{M}$ constant	$4L$

Using these values for SVRG, the number of iterations in the inner loop are

$$\begin{aligned} T &\geq (n+2)\log(640) \\ S\log(k+1) &\geq \max\left\{(2n+1)\log\left(32 \cdot \frac{n}{L}\left(\frac{L}{n} + L\right) \cdot (k+1)^2\right)\right\}. \end{aligned}$$

The global complexity for SVRG when n is sufficiently large is

1. Algorithm 10 will generate a point x satisfying $\operatorname{dist}(0, \partial f(x)) \leq \varepsilon$ after at most

$$O\left[n \log\left(\frac{L}{\kappa_0}\right) \log\left(\frac{nL^2(f(x_0) - f^*)^2}{\varepsilon^4}\right) \cdot \left(\frac{L(f(x_0) - f^*)}{\varepsilon^2}\right)\right]$$

gradient computations.

2. If f is convex, then Algorithm 10 will generate a point x satisfying $\operatorname{dist}(0, \partial f(x)) \leq \varepsilon$ after at most

$$O\left[n^{2/3} \log\left(\frac{L}{\kappa_0}\right) \log\left(\frac{L^{4/3} n\left\|x^* - x_0\right\|^{4/3}}{\varepsilon^{4/3}}\right) \cdot \left(\frac{L^{2/3}\left\|x^* - x_0\right\|^{2/3}}{\varepsilon^{2/3}}\right)\right]$$

gradient computations.

3. If f is convex, then Algorithm 10 will generate a point x satisfying $f(x) - f^* \leq \varepsilon$ after at most

$$O\left[\sqrt{n}\log\left(\frac{L}{\kappa_0}\right)\cdot\log\left(\frac{nL\left\|x_0 - x^*\right\|^2}{\varepsilon}\right)\cdot\frac{\sqrt{L}\left\|x^* - x_0\right\|}{\sqrt{\varepsilon}}\right]$$

gradient computations.

For SAGA, we have the following table of values

Variable	Description	Value
$1/\tau_L$	linear convergence with $\kappa = L$	$n + \sqrt{2n}$
κ_{cvx}	smooth parameter for convex setting	$L/(2n+1)$
$1/\tau_{\kappa_{\text{cvx}}}$	linear convergence with $\kappa_{\text{cvx}} = L/(2n+1)$	$n + \sqrt{2n^2 + n + 1}$
A_{4L}	Method $\mathcal{M}$ constant	$4L$

We observe that the variables for SAGA are only constant multiples as those for SVRG. Therefore, the global complexities results for SAGA are, up to constants, the same as SVRG.

We present a summary of common choices for the parameters when minimizing the composite form, $\frac{1}{n}\sum_{i=1}^n f_i(x) + \psi(x)$ where L is the Lipschitz constant.

Variable	Description	Initialization
$x_0 \in \mathbb{R}^p$	Starting point; $x_0 \in \text{dom}(f)$	Random
$\kappa_0 > 0$	Moreau envelope parameter for proximal term; (Smallest) initial guess for ρ	$2L/n$
$\kappa_{\text{cvx}} > 0$	Moreau envelope parameter for acceleration term	$2L/n$
T	Number of iterations to produce $\bar{x}_k$	n
S	Number of iterations to produce $\tilde{x}_k$	n

D.6 Experiments

We investigate the performance of 4WD-Catalyst in two standard non-convex problems in machine learning. We report experimental results of 4WD-Catalyst when applied to two different algorithms: SVRG [Xiao and Zhang, 2014] and SAGA [Defazio et al., 2014a]. We compare the following algorithms:

- Nonconvex SVRG/SAGA [Reddi et al., 2016b]: stepsize $\eta = 1/Ln^{2/3}$
- Convex SVRG/SAGA [Defazio et al., 2014a, Xiao and Zhang, 2014]: stepsize $\eta = 1/2L$
- 4WD-Catalyst SVRG/SAGA: stepsize $\eta = 1/2L$

The original version of SVRG (resp. SAGA), convex SVRG (resp. SAGA), was designed for minimizing convex objectives. We report their results, while there is no theoretical guarantee on their behavior when venturing into nonconvex terrains. We also report the results of recently proposed variants, Nonconvex SVRG/SAGA, designed for minimizing nonconvex objectives. The proposed algorithms 4WD-Catalyst SVRG and 4WD-Catalyst SAGA enjoy the strong theoretical guarantees stated in Sec. 3.

Parameter settings We start from an initial estimate of the Lipschitz constant L and use the theoretically recommended $\kappa_0 = \kappa_{\text{cvx}} = 2L/n$. The number of inner iterations is to $T = S = n$ in all experiments, which boils down to making one pass at most over the data for solving each sub-problem. We simply drop the $\log(k)$ dependency while solving the subproblem in (D.17). These choices turn out to be justified *a posteriori*, as both SVRG and SAGA have a much better convergence rate in practice than the theoretical rate derived from a worst-case analysis. Indeed, in all experiments, one pass over the data to solve each sub-problem is enough to guarantee sufficient descent.

Sparse matrix factorization a.k.a. dictionary learning. Dictionary learning consists of representing a dataset $X = [x_1, \cdots, x_n] \in \mathbb{R}^{m\times n}$ as a product $X \approx DA$, where D in $\mathbb{R}^{m\times p}$ is called a dictionary, and A in $\mathbb{R}^{p\times n}$ is a sparse matrix. The classical non-convex formulation [see Mairal et al., 2014] is

$$\min_{D\in\mathcal{C}, A\in\mathbb{R}^{p\times n}} \sum_{i=1}^{n} \frac{1}{2}\|x_i - D\alpha_i\|_2^2 + \psi(\alpha_i),$$

where $A = [\alpha_1 \cdots \alpha_n]$ carries the decomposition coefficients of signals $x_1 \cdots x_n$, ψ is a sparsity-inducing regularization and $\mathcal{C}$ is chosen as the set of matrices whose columns are in the ℓ_2-ball. An equivalent point of view is the finite-sum problem $\min_{D\in\mathcal{C}} \frac{1}{n}\sum_{i=1}^{n} f_i(D)$ with

$$f_i(D) := \min_{\alpha\in\mathbb{R}^p} \frac{1}{2}\|x_i - D\alpha\|_2^2 + \psi(\alpha). \tag{D.22}$$

We consider elastic-net regularization $\psi(\alpha) = \frac{\mu}{2}\|\alpha\|^2 + \lambda\|\alpha\|_1$ of [Zou and Hastie, 2005], which has a sparsity-inducing effect, and report the corresponding results in Figure D.1 and D.2,

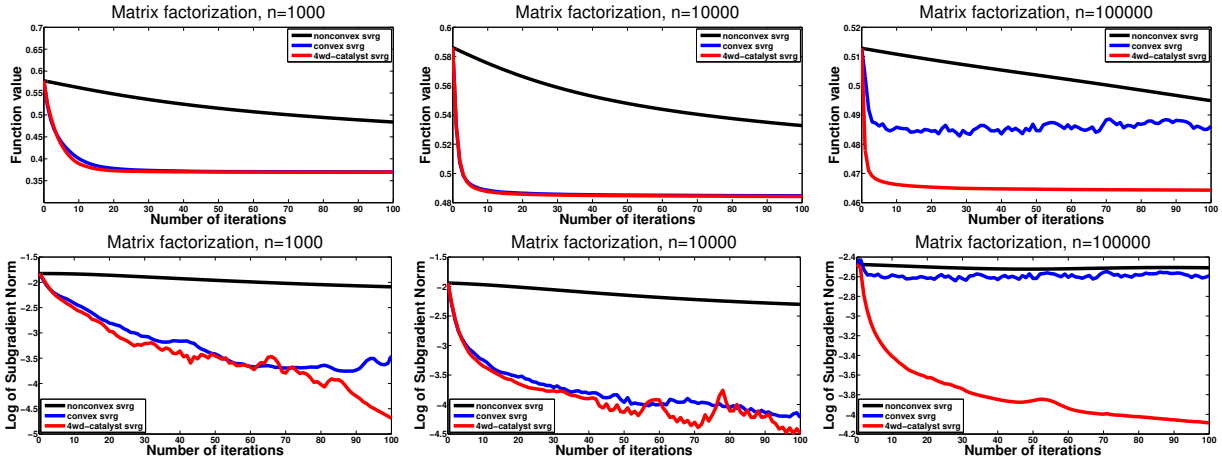

Figure D.1 – Dictionary learning experiments using SVRG. We plot the function value (top) and the subgradient norm (bottom). From left to right, we vary the size of dataset from $n = 1\,000$ to $n = 100\,000$.

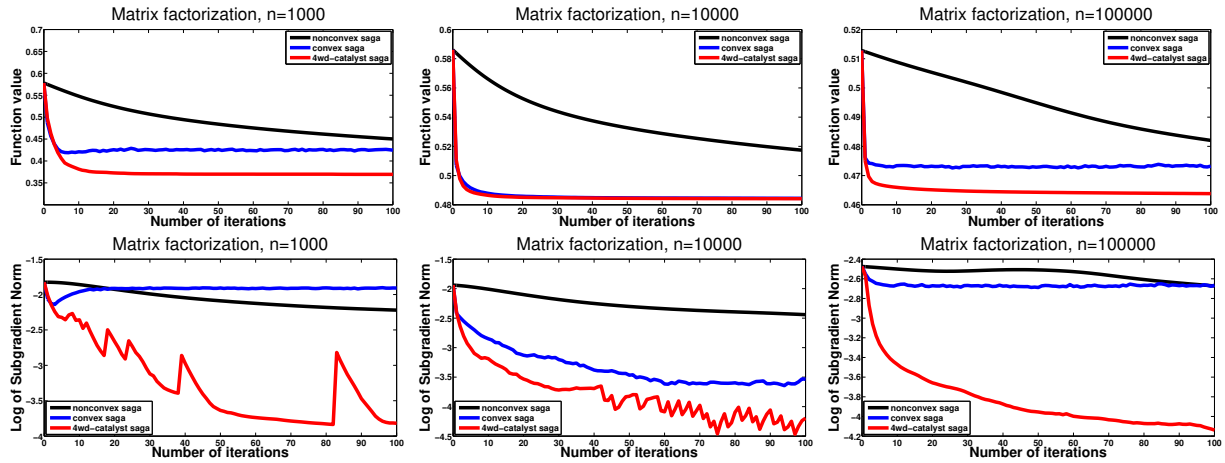

Figure D.2 – Dictionary learning experiments using SAGA. We plot the function value (top) and the subgradient norm (bottom). From left to right, we vary the size of dataset from $n = 1\,000$ to $n = 100\,000$.

learning a dictionary in $\mathbb{R}^{m\times p}$ with $p = 256$ elements, on a set of whitened normalized image patches of size $m = 8\times 8$. Parameters are standard ones in this literature [Mairal et al., 2014]—that is, a small value $\mu = 1e-5$, and $\lambda = 0.25$, leading to sparse matrices A (on average ≈ 4 non-zero coefficients per column of A).

Neural networks. We consider now simple binary classification problems for learning neural networks. Assume that we are given a training set $\{a_i, b_i\}_{i=1}^n$, where the variables b_i in $\{-1, +1\}$

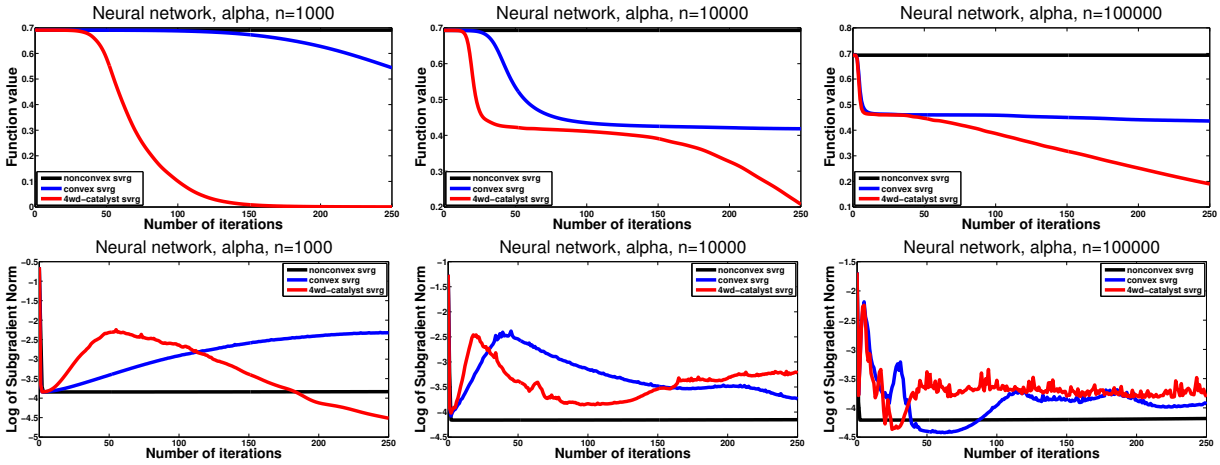

Figure D.3 – Neural network experiments on subsets of dataset alpha. From left to right, we vary the size of the dataset's subset from $n = 1\,000$ to $n = 100\,000$.

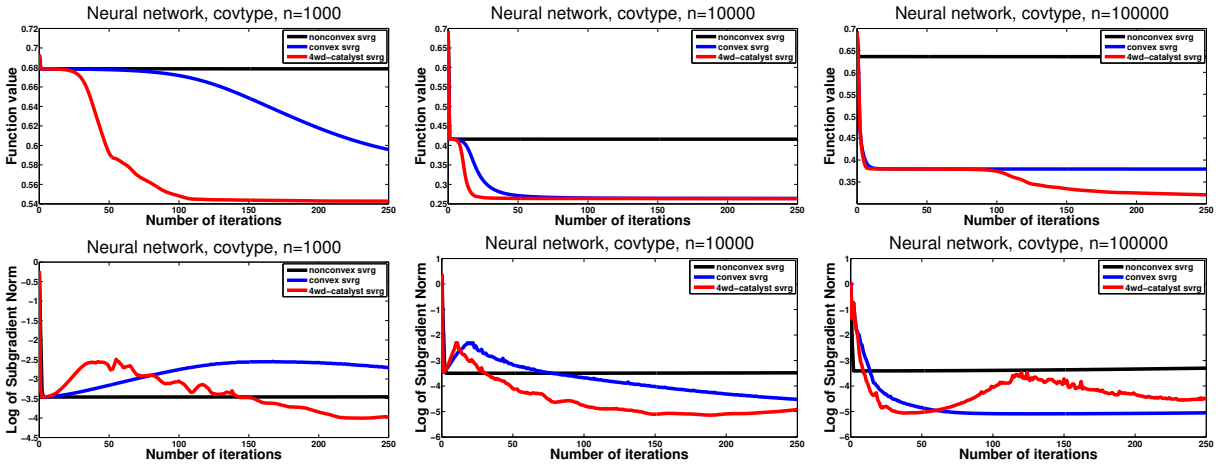

Figure D.4 – Neural network experiments on subsets of datasets alpha (top) and covtype (bottom).

represent class labels, and a_i in $\mathbb{R}^p$ are feature vectors. The estimator of a label class is now given by a two-layer neural network $\hat{b} = \text{sign}(W_2^\top \sigma(W_1^\top a))$, where W_1 in $\mathbb{R}^{p\times d}$ represents the weights of a hidden layer with d neurons, W_2 in $\mathbb{R}^d$ carries the weight of the network's second layer, and $\sigma(u) = \log(1 + e^u)$ is a non-linear function, applied pointwise to its arguments. We fix the number of hidden neurons to $d = 100$ and use the logistic loss to fit the estimators to the true labels. Since the memory required by SAGA becomes n times larger than SVRG for nonlinear models, which is problematic for large n, we can only perform experiments with SVRG. The experimental results are reported on two datasets alpha and covtype in Figure D.3 and D.4.

Initial estimates of L. The proposed algorithm 4WD-Catalyst requires an initial estimate of the Lipschitz constant L. In the problems we are considering, there is no simple closed form formula available to compute an estimate of L. We use following heuristics to estimate L:

1. For matrix factorization, it can be shown that the function f_i defined in (D.22) is differentiable according to Danskin's theorem [see Bertsekas [Bertsekas, 1999], Proposition B.25] and its gradient is given by

$$\nabla_D f_i(D) = -(x_i - D\alpha_i(D))\alpha_i(D)^T \quad \text{where} \quad \alpha_i(D) \in \arg\min_{\alpha \in \mathbb{R}^p} \frac{1}{2} \|x_i - D\alpha\|^2 + \psi(\alpha).$$

   If $\alpha_i(D)$ is in fact D-independant, the gradient is linear in D and thus admit $\|\alpha_i\|^2$ as Lipschitz constant. Thus when initializing our algorithm at D_0, we use a Lasso/Elastic-net solver spams to find $\alpha_i(D_0)$ for any $i \in [1, n]$ and use $\max_{i\in[1,n]} \|\alpha_i(D_0)\|^2$ as an estimate of L.

2. For neural network, the formulation we are considering is actually differentiable. We randomly generates two pairs of weight vectors (W_1, W_2) and (W_1', W_2') and use the quantity

$$\max_{i\in[1,n]} \left\{ \frac{\|\nabla f_i(W_1, W_2) - \nabla f_i(W_1', W_2)\|}{\|W_1 - W_1'\|}, \frac{\|\nabla f_i(W_1, W_2) - \nabla f_i(W_1, W_2')\|}{\|W_2 - W_2'\|} \right\}$$

   as an estimate of the Lipschitz constant, where f_i denotes the loss function respect to i-th training sample (a_i, b_i). We separate weights in each layer to estimate the Lipschitz constant *per layer*. Indeed the scales of the weights can be quite different across layers.

Computational cost. For the Convex-SVRG and Nonconvex-SVRG, one iteration corresponds to one pass over the data in the plots. On the one hand, since 4WD-Catalyst-SVRG solves two sub-problems per iteration, the cost per iteration is twice that of the Convex-SVRG and Nonconvex-SVRG. On the other hand, in the experiments, we observe that, every time acceleration occurs, then $\tilde{x}_k$ is almost always preferred to $\bar{x}_k$ in step 4 of 4WD-Catalyst, hence half of the computations are in fact not performed when running 4WD-Catalyst-SVRG.

We report in Figure D.5 an experimental study where we vary S on the neural network example. In terms of number of iterations, of course, the larger S_k the better the performance. This is not surprising as we solve each subproblem more accurately. Nevertheless, in terms of number of gradient evaluations, the relative performance is reversed. There is clearly no benefit to take larger S_k. This justifies in hindsight our choice of setting $S = 1$.

Experimental conclusions. In matrix factorization experiments, we observe that 4WD-Catalyst-SVRG always outperforms the competing algorithms. Nonconvex-SVRG has slower convergence in objective values and Convex-SVRG is not always converging; see in particular right panel in Fig. D.1. Therefore 4WD-Catalyst-SVRG offers a more stable option than

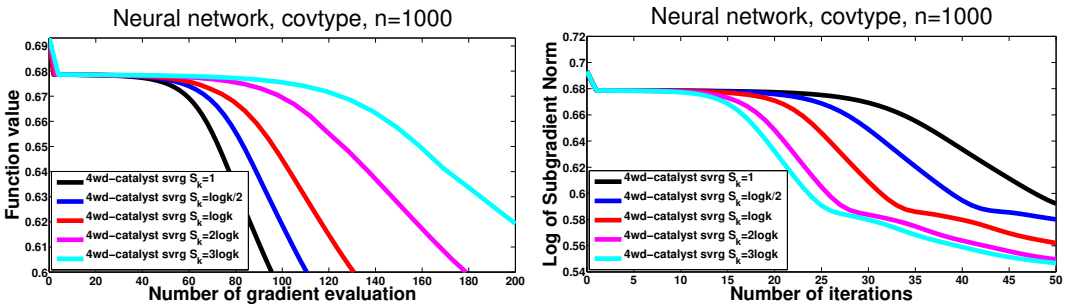

Figure D.5 – We run 50 iterations of 4WD-Catalyst SVRG with different choice of S on two-layer neural network. The data is a subset of dataset covtype. The x-axis is the number of gradient evaluations on the left, which is $T + S_k$ per iteration with $T = 1$; and the number of iterations on the right.

Convex-SVRG for minimizing nonconvex objectives. Furthermore, in these experiments 4WD-Catalyst-SVRG enjoys a faster convergence in objective values. This confirms the remarkable ability of 4WD-Catalyst-SVRG to adapt to nonconvex terrains. Similar conclusions hold when applying 4WD-Catalyst to SAGA; see Sec.6. in Appendix, which demonstrates how general 4WD-Catalyst is.

In neural network experiments, we observe that 4WD-Catalyst-SVRG converges much faster in terms of objective values than the competing algorithms. Nonconvex-SVRG with the theoretically recommended sequence of step-sizes [Reddi et al., 2016b] compares unfavorably here, which implies that the recommended step-sizes are too pessimistic hence too small. We also observe an interesting phenomenon: the subgradient norm may increase at some point then decrease, while the function value keeps decreasing, as the algorithm proceeds. This suggests that the extrapolation step, or the Auto-adapt procedure, is helpful to escape bad stationary points, *e.g.*, saddle-points. A more systematic study is required to confirm such observation, we leave it as a potential direction of future work.

D.7 Note on convergence rates in strongly-convex composite minimization

We now briefly discuss convergence rates, which are typically given in different forms in the convex and non-convex cases. If the weak-convex constant is known, we can form a strongly convex approximation similar to [Lin et al., 2015]. For that purpose, we consider a strongly-convex composite minimization problem

$$\min_{x \in \mathbb{R}^p} \; h(x) := f_0(x) + \psi(x),$$

where $f_0 \colon \mathbb{R}^p \to \mathbb{R}$ is μ-strongly convex and smooth with L-Lipschitz continuous gradient ∇f_0, and $\psi \colon \mathbb{R}^p \to \overline{\mathbb{R}}$ is a closed convex function with a computable proximal map

$$\operatorname{prox}_{\beta\psi}(y) := \operatorname*{arg\,min}_{z \in \mathbb{R}^p} \left\{ \psi(y) + \tfrac{1}{2\beta}\|z - y\|^2 \right\}.$$

Let x^* be the minimizer of h and h^* be the minimal value of h. In general, there are three types of measures of optimality that one can monitor: $\|x - x^*\|^2$, $h(x) - h^*$, and $\operatorname{dist}(0, \partial h(x))$.

Since h is strongly convex, the three of them are equivalent in terms of convergence rates if one can take an extra *prox-gradient step*:

$$[x]_L := \operatorname{prox}_{\psi/L}(x - L^{-1}\nabla f_0(x)).$$

To see this, define the *displacement vector*, also known as the gradient mapping, $g_L(x) := L(x - [x]_L)$, and notice the inclusion $g_L(x) \in \partial h([x]_L)$. In particular $g_L(x) = 0$ if and only if x is the minimizer of h. These next inequalities follow directly from Theorem 2.2.7 in [Nesterov, 2004]:

$$\begin{aligned}
\tfrac{1}{2L}\|g_L(x)\| \leq &\|x - x^*\| \leq \tfrac{2}{\mu}\|g_L(x)\| \\
\tfrac{\mu}{2}\|x - x^*\|^2 \leq &h(x) - h^* \leq \tfrac{1}{2\mu}|\partial h(x)|^2 \\
2\mu(h([x]_L) - h^*) \leq &\|g_L(x)\|^2 \leq 2L(h(x) - h([x]_L))
\end{aligned}$$

Thus, an estimate of any one of the four quantities $\|x - x^*\|$, $h(x) - h^*$, $\|g_L(x)\|$, or $\operatorname{dist}(0, \partial h(x))$ directly implies an estimate of the other three evaluated either at x or at $[x]_L$.

D.8 Theoretical analysis of the basic algorithm

We present here proofs of the theoretical results of the paper. Althroughout the proofs, we shall work under the Assumptions on f stated in Section D.3 and the Assumptions on $\mathcal{M}$ stated in Section D.4.

D.8.1 Convergence guarantee of Basic 4WD-Catalyst

In Theorem 12 and Theorem 13 under an appropriate tolerance policy on the proximal subproblems (D.5) and (D.7), Basic 4WD-Catalyst performs no worse than an exact proximal point method in general, while automatically accelerating when f is convex. For this, we need the following observations.

Lemma 22 (Growth of (α_k))**.** *Suppose the sequence $\{\alpha_k\}_{k \geq 1}$ is produced by Algorithm 9. Then, the following bounds hold for all $k \geq 1$:*

$$\frac{\sqrt{2}}{k+2} \leq \alpha_k \leq \frac{2}{k+1}.$$

Proof. This result is noted without proof in a remark of [Tseng, 2008]. For completeness, we give below a simple proof using induction. Clearly, the statement holds for $k = 1$. Assume the inequality on the right-hand side holds for k. By using the induction hypothesis, we get

$$\alpha_{k+1} = \frac{\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2}{2} = \frac{2}{\sqrt{1 + 4/\alpha_k^2} + 1} \leq \frac{2}{\sqrt{1 + (k+1)^2} + 1} \leq \frac{2}{k+2},$$

as claimed and the expression for α_{k+1} is given by explicitly solving (D.9). To show the lower bound, we note that for all $k \geq 1$, we have

$$\alpha_{k+1}^2 = (1 - \alpha_{k+1})\alpha_k^2 = \prod_{i=2}^{k+1}(1 - \alpha_i)\alpha_1^2 = \prod_{i=2}^{k+1}(1 - \alpha_i).$$

Using the established upper bound $\alpha_k \leq \frac{2}{k+1}$ yields

$$\alpha_{k+1}^2 \geq \prod_{i=2}^{k+1}\left(1 - \frac{2}{i+1}\right) = \frac{2}{(k+2)(k+1)} \geq \frac{2}{(k+2)^2}.$$

The result follows. □

Lemma 23 (Prox-gradient and near-stationarity)**.** *Suppose y^+ satisfies dist$(0, \partial f_\kappa(y^+; y)) < \varepsilon$. Then, the inequality holds:*

$$\operatorname{dist}\big(0, \partial f(y^+)\big) \leq \varepsilon + \big\|\kappa(y^+ - y)\big\|.$$

Proof. We can find $\xi \in \partial f_\kappa(y^+; y)$ with $\|\xi\| \leq \varepsilon$. Taking into account $\partial f_\kappa(y^+; y) = \partial f(y^+) + \kappa(y^+ - y)$ the result follows. □

Next we establish convergence guarantees of Theorem 12 and Theorem 13 for Basic 4WD-Catalyst .

Proof of Theorem 13 and Theorem 13. The proof of Theorem 12 follows the analysis of inexact proximal point method [Lin et al., 2015, Güler, 1991, Bertsekas, 2015]. The descent condition in (D.11) implies $\{f(x_k)\}_{k\geq 0}$ are monotonically decreasing. From this, we deduce

$$f(x_{k-1}) = f_\kappa(x_{k-1}; x_{k-1}) \geq f_\kappa(\bar{x}_k; x_{k-1}) \geq f(x_k) + \frac{\kappa}{2}\|\bar{x}_k - x_{k-1}\|^2. \tag{D.23}$$

Using the adaptive stationarity condition (D.11), we apply Lemma 23 with $y = x_{k-1}$, $y^+ = \bar{x}_k$ and $\varepsilon = \kappa\|\bar{x}_k - x_{k-1}\|$; hence we obtain

$$\operatorname{dist}(0, \partial f(\bar{x}_k)) \leq 2\|\kappa(\bar{x}_k - x_{k-1})\|.$$

We combine the above inequality with (D.23) to deduce

$$\operatorname{dist}^2(0, \partial f(\bar{x}_k)) \leq 4 \left\| \kappa(\bar{x}_k - x_{k-1}) \right\|^2 \leq 8\kappa \left(f(x_{k-1}) - f(x_k) \right). \tag{D.24}$$

Summing $j = 1$ to N, we conclude

$$\begin{aligned}
\min_{j=1,\ldots,N} \left\{ \operatorname{dist}^2(0, \partial f(\bar{x}_j)) \right\} &\leq \frac{4}{N} \sum_{j=1}^{N} \left\| \kappa(\bar{x}_k - x_{k-1}) \right\|^2) \\
&\leq \frac{8\kappa}{N} \left(\sum_{j=1}^{N} f(x_{j-1}) - f(x_j) \right) \\
&\leq \frac{8\kappa}{N} \left(f(x_0) - f^* \right).
\end{aligned}$$

Next, suppose the function f is convex. Our analysis is similar to that of [Tseng, 2008, Beck and Teboulle, 2009]. Using the stopping criteria (D.12), fix an $\xi_k \in \partial f_\kappa(\tilde{x}_k; y_k)$ with $\|\xi_k\| < \frac{\kappa}{k+1} \|\tilde{x}_k - y_k\|$. For any $x \in \mathbb{R}^n$, Equation (D.10), and the strong convexity of the function $f_\kappa(\cdot; y_k)$ yields

$$f(x_k) \leq f(\tilde{x}_k) \leq f(x) + \frac{\kappa}{2} \left(\|x - y_k\|^2 - \|x - \tilde{x}_k\|^2 - \|\tilde{x}_k - y_k\|^2 \right) + \xi_k^T (\tilde{x}_k - x).$$

We substitute $x = \alpha_k x^* + (1 - \alpha_k) x_{k-1}$ where x^* is any minimizer of f. Using the convexity of f, the norm of ξ_k, and Equations (D.6) and (D.8), we deduce

$$\begin{aligned}
f(x_k) \leq \alpha_k f(x^*) + (1 - \alpha_k) f(x_{k-1}) + \frac{\alpha_k^2 \kappa}{2} \left(\|x^* - v_{k-1}\|^2 - \|x^* - v_k\|^2 \right) \\
- \frac{\kappa}{2} \|\tilde{x}_k - y_k\|^2 + \frac{\alpha_k \kappa}{k+1} \|\tilde{x}_k - y_k\| \|x^* - v_k\|.
\end{aligned} \tag{D.25}$$

Set $\theta_k = \frac{1}{k+1}$. Completing the square on Equation (D.25), we obtain

$$\frac{-\kappa}{2} \|\tilde{x}_k - y_k\|^2 + \alpha_k \theta_k \kappa \|\tilde{x}_k - y_k\| \|x^* - v_k\| \leq \frac{\kappa}{2} (\alpha_k \theta_k)^2 \|x^* - v_k\|^2.$$

Hence, we deduce

$$\begin{aligned}
f(x_k) - f^* &\leq (1 - \alpha_k)(f(x_{k-1}) - f^*) + \frac{\alpha_k^2 \kappa}{2} \left(\|x^* - v_{k-1}\|^2 - \|x^* - v_k\|^2 \right) \\
&\quad + \frac{\kappa}{2} (\alpha_k \theta_k)^2 \|x^* - v_k\|^2. \\
&= (1 - \alpha_k)(f(x_{k-1}) - f^*) + \frac{\alpha_k^2 \kappa}{2} \left(\|x^* - v_{k-1}\|^2 - \left(1 - \theta_k^2\right) \|x^* - v_k\|^2 \right)
\end{aligned}$$

Denote $A_k := 1 - \theta_k^2$. Subtracting f^* from both sides and using the inequality $\frac{1-\alpha_k}{\alpha_k^2} = \frac{1}{\alpha_{k-1}^2}$ and $\alpha_1 \equiv 1$, we derive the following recursion argument:

$$\begin{aligned}\frac{f(x_k) - f^*}{\alpha_k^2} + \frac{A_k\kappa}{2}\|x^* - v_k\|^2 &\leq \frac{1-\alpha_k}{\alpha_k^2}\left(f(x_{k-1}) - f^*\right) + \frac{\kappa}{2}\|x^* - v_{k-1}\|^2 \\ &\leq \frac{1}{A_{k-1}}\left(\frac{f(x_{k-1}) - f^*}{\alpha_{k-1}^2} + \frac{A_{k-1}\kappa}{2}\|x^* - v_{k-1}\|^2\right).\end{aligned}$$

The last inequality follows because $0 < A_{k-1} \leq 1$. Iterating N times,we deduce

$$\frac{f(x_N) - f^*}{\alpha_N^2} \leq \prod_{j=2}^{N} \frac{1}{A_{j-1}}\left(\frac{\kappa}{2}\|x^* - v_0\|^2\right). \tag{D.26}$$

We note

$$\prod_{j=2}^{N} \frac{1}{A_{j-1}} = \frac{1}{\prod_{j=2}^{N}\left(1 - \frac{1}{(j+1)^2}\right)} \leq 2; \tag{D.27}$$

thereby concluding the result. Summing up (D.24) from $j = N+1$ to $2N$, we obtain

$$\begin{aligned}\min_{j=1,\ldots,2N}\left\{\mathrm{dist}^2(0, \partial f(\bar{x}_j))\right\} &\leq \frac{4}{N}\sum_{j=N+1}^{2N}\|\kappa(\bar{x}_k - x_{k-1})\|^2) \\ &\leq \frac{8\kappa}{N}\left(\sum_{j=N+1}^{2N} f(x_{j-1}) - f(x_j)\right) \\ &\leq \frac{8\kappa}{N}\left(f(x_N) - f^*\right)\end{aligned}$$

Combining this inequality with (D.26), the result is shown.

□

D.9 Analysis of 4WD-Catalyst and Auto-adapt

Linear convergence interlude. Our assumption on the linear rate of convergence of $\mathcal{M}$ (see (D.14)) may look strange at first sight. Nevertheless, most linearly convergent first-order methods $\mathcal{M}$ for composite minimization either already satisfy this assumption or can be made to satisfy it by introducing an extra prox-gradient step. To see this, recall the convex composite minimization problem from Section D.7

$$\min_{z\in\mathbb{R}^p} h(z) := f_0(z) + \psi(z),$$

where

1. $f_0 \colon \mathbb{R}^p \to \mathbb{R}$ is convex and C^1-smooth with the gradient ∇f_0 that is L-Lipschitz,
2. $\psi \colon \mathbb{R}^p \to \overline{\mathbb{R}}$ is a closed convex function with a computable proximal map

$$\operatorname{prox}_{\beta\psi}(y) := \arg\min_z \{\psi(y) + \tfrac{1}{2\beta}\|z - y\|^2\}.$$

See [Parikh and Boyd, 2014] for a survey of proximal maps. Typical linear convergence guarantees of an optimization algorithm assert existence of constants $A \in \mathbb{R}$ and $\tau \in (0, 1)$ satisfying

$$h(z_t) - h^* \leq A(1 - \tau)^t (h(z_0) - h^*) \tag{D.28}$$

for each $t = 0, 1, 2, \ldots, \infty$. To bring such convergence guarantees into the desired form (D.14), define the prox-gradient step

$$[z]_L := \operatorname{prox}_{\psi/L}(z - L^{-1}\nabla f_0(z)),$$

and the displacement vector

$$g_L(z) = L(z - [z]_L),$$

and notice the inclusion $g_L(z) \in \partial h([z]_L)$. The following inequality follows from [Nesterov, 2013]:

$$\|g_L(z)\|^2 \leq 2L(h(z) - h([z]_L)) \leq 2L(h(z) - h^*).$$

Thus, the linear rate of convergence (D.28) implies

$$\|g_L(z_t)\|^2 \leq 2LA(1 - \tau)^t (h(z_0) - h^*),$$

which is exactly in the desired form (D.14).

D.9.1 Convergence analysis of the adaptive algorithm: 4WD-Catalyst

First, under some reasonable assumptions on the method $\mathcal{M}$ (see Section D.4), the sub-method Auto-adapt terminates.

Lemma 24 (Auto-adapt terminates)**.** *Assume that $\tau_\kappa \to 1$ when $\kappa \to +\infty$. The procedure Auto-adapt$(x, \kappa, \varepsilon, T)$ terminates after finitely many iterations.*

Proof. Due to our assumptions on $\mathcal{M}$ and the expressions $f_\kappa(x; x) = f(x)$ and $f_\kappa^*(x) \geq f^*$, we have

$$\operatorname{dist}^2\Big(0, \partial f_\kappa(z_T; x)\Big) \leq A(1 - \tau_\kappa)^T \Big(f(x) - f_\kappa^*(x)\Big) \leq A(1 - \tau_\kappa)^T \Big(f(x) - f^*\Big). \tag{D.29}$$

Since τ_κ tends to one, for all sufficiency large κ, we can be sure that the right-hand-side is smaller than ε^2. On the other hand, for $\kappa > \rho$, the function $f_\kappa(\cdot; x)$ is $(\kappa - \rho)$-strongly convex and therefore we have $\operatorname{dist}^2(0, \partial f_\kappa(z_T; x)) \geq 2(\kappa - \rho)(f_\kappa(z_T; x) - f_\kappa^*(x))$. Combining this with (D.29), we deduce

$$f_\kappa(z_T; x) - f_\kappa^*(x) \leq \frac{A(1 - \tau_\kappa)^T}{2(\kappa - \rho)} \Big(f(x) - f_\kappa^*(x) \Big).$$

Letting $\kappa \to \infty$, we deduce $f_\kappa(z_T; x) \leq f(x)$, as required. Thus the loop indeed terminates. □

We prove the main result, Theorem 15, for 4WD-Catalyst.

Proof of Theorem 15. The proof closely resembles the proofs of Theorem 13 and Theorem 13, so we omit some of the details. The main difference in the proof is that we keep track of the effects the parameters κ_{cvx} and κ_0 have on the inequalities as well as the sequence of κ_k. Since $\{f(x_k)\}_{k \geq 0}$ are monotonically decreasing, we deduce

$$f(x_{k-1}) = f_{\kappa_k}(x_{k-1}; x_{k-1}) \geq f_{\kappa_k}(\bar{x}_k; x_{k-1}) \geq f(x_k) + \frac{\kappa_k}{2} \left\| \bar{x}_k - x_{k-1} \right\|^2. \tag{D.30}$$

Using the adaptive stationary condition (D.18), we apply Lemma 23 with $\varepsilon = \kappa_k \left\| \bar{x}_k - x_{k-1} \right\|$; hence we obtain

$$\operatorname{dist}(0, \partial f(\bar{x}_k)) \leq 2 \left\| \kappa_k (\bar{x}_k - x_{k-1}) \right\|.$$

We combine the above inequality with (D.30) to deduce

$$\operatorname{dist}^2(0, \partial f(\bar{x}_k)) \leq 4 \left\| \kappa_k (\bar{x}_k - x_{k-1}) \right\|^2 \leq 8\kappa_{\max} \left(f(x_{k-1}) - f(x_k) \right). \tag{D.31}$$

Summing $j = 1$ to N, we conclude

$$\begin{aligned} \min_{j=1,\ldots,N} \left\{ \operatorname{dist}^2(0, \partial f(\bar{x}_j)) \right\} &\leq \frac{4}{N} \sum_{j=1}^{N} 2 \left\| \kappa_k (\bar{x}_k - x_{k-1}) \right\|^2) \\ &\leq \frac{8\kappa_{\max}}{N} \left(\sum_{j=1}^{N} f(x_{j-1}) - f(x_j) \right) \\ &\leq \frac{8\kappa_{\max}}{N} \left(f(x_0) - f^* \right). \end{aligned}$$

Suppose the function f is convex. Using in the stopping criteria (D.19) in replacement of (D.11), we deduce a similar expression as (D.25):

$$\begin{aligned} f(x_k) \leq \alpha_k f(x^*) + (1 - \alpha_k) f(x_{k-1}) + \frac{\alpha_k^2 \kappa_{\text{cvx}}}{2} \left(\left\| x^* - v_{k-1} \right\|^2 - \left\| x^* - v_k \right\|^2 \right) \\ - \frac{\kappa_{\text{cvx}}}{2} \left\| \tilde{x}_k - y_k \right\|^2 + \frac{\alpha_k \kappa_{\text{cvx}}}{k+1} \left\| \tilde{x}_k - y_k \right\| \left\| x^* - v_k \right\|. \end{aligned}$$

Denote $\theta_k = \frac{1}{k+1}$. Completing the square, we obtain

$$\frac{-\kappa_{\text{cvx}}}{2}\left\|\tilde{x}_k - y_k\right\|^2 + \alpha_k\theta_k\kappa_{\text{cvx}}\left\|\tilde{x}_k - y_k\right\|\left\|x^* - v_k\right\| \leq \frac{\kappa_{\text{cvx}}}{2}\left(\alpha_k\theta_k\right)^2\left\|x^* - v_k\right\|^2.$$

Hence, we deduce

$$\begin{aligned} f(x_k) - f^* &\leq (1-\alpha_k)(f(x_{k-1}) - f^*) + \frac{\alpha_k^2\kappa_{\text{cvx}}}{2}\left(\left\|x^* - v_{k-1}\right\|^2 - \left\|x^* - v_k\right\|^2\right) \\ &\quad + \frac{\kappa_{\text{cvx}}}{2}\left(\alpha_k\theta_k\right)^2\left\|x^* - v_k\right\|^2. \\ &= (1-\alpha_k)(f(x_{k-1}) - f^*) + \frac{\alpha_k^2\kappa_{\text{cvx}}}{2}\left(\left\|x^* - v_{k-1}\right\|^2 - \left(1-\theta_k^2\right)\left\|x^* - v_k\right\|^2\right) \end{aligned}$$

Denote $A_k := 1-\theta_k^2$. Following the standard recursion argument as in the proofs of Theorem 13 and Theorem 13, we conclude

$$\begin{aligned} \frac{f(x_k) - f^*}{\alpha_k^2} + \frac{A_k\kappa_{\text{cvx}}}{2}\left\|x^* - v_k\right\|^2 &\leq \frac{1-\alpha_k}{\alpha_k^2}\left(f(x_{k-1}) - f^*\right) + \frac{\kappa_{\text{cvx}}}{2}\left\|x^* - v_{k-1}\right\|^2 \\ &\leq \frac{1}{A_{k-1}}\left(\frac{f(x_{k-1}) - f^*}{\alpha_{k-1}^2} + \frac{A_{k-1}\kappa_{\text{cvx}}}{2}\left\|x^* - v_{k-1}\right\|^2\right). \end{aligned}$$

The last inequality follows because $0 < A_{k-1} \leq 1$. Iterating N times, we deduce

$$\frac{f(x_N) - f^*}{\alpha_N^2} \leq \prod_{j=2}^{N}\frac{1}{A_{j-1}}\left(\frac{\kappa_{\text{cvx}}}{2}\left\|x^* - v_0\right\|^2\right). \tag{D.32}$$

We note

$$\prod_{j=2}^{N}\frac{1}{A_{j-1}} = \frac{1}{\prod_{j=2}^{N}\left(1 - \frac{1}{(j+1)^2}\right)} = 2;$$

thus the result is shown. Summing up (D.31) from $j = N+1$ to $2N$, we obtain

$$\begin{aligned} \min_{j=1,\ldots,2N}\left\{\text{dist}^2(0, \partial f(\bar{x}_j))\right\} &\leq \frac{1}{N}\sum_{j=N+1}^{2N} 2\left(1 + \frac{\kappa_0^2}{\kappa_k^2}\right)\left\|\kappa_k(\bar{x}_k - x_{k-1})\right\|^2) \\ &\leq \frac{8\kappa_{\max}}{N}\left(\sum_{j=N+1}^{2N} f(x_{j-1}) - f(x_j)\right) \\ &\leq \frac{8\kappa_{\max}}{N}\left(f(x_N) - f^*\right) \end{aligned}$$

Combining this inequality with (D.32), the result is shown. □

D.10 Inner-loop complexity: proof of Theorem 16

Recall, the following notation

$$f_0(x;y) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \frac{\kappa}{2}\|x-y\|^2$$
$$y^0 = \text{prox}_{1/(\kappa+L)f_0}\left(y - \frac{1}{\kappa+L}\nabla f_0(y;y)\right). \tag{D.33}$$

Lemma 25 (Relationship between function values and iterates of the prox)**.** *Assuming $\psi(x)$ is convex and the parameter $\kappa > \rho$, then*

$$f_\kappa(y^0;y) - f_\kappa^*(y) \leq \frac{\kappa+L}{2}\|y^* - y\|^2 \tag{D.34}$$

where y^* is a minima of $f_\kappa(\cdot;y)$ and $f_\kappa^*(y)$ is the optimal value.

Proof. As the κ is chosen sufficiently large, we know $f_0(\cdot;y)$ is convex and differentiable with $(\kappa+L)$-Lipschitz continuous gradient. Hence, we deduce for all x

$$f_0(y;y) + \nabla f_0(y;y)^T(x-y) \leq f_0(x;y). \tag{D.35}$$

Using the definition of y^0 and the $(\kappa-L)$-Lip. continuous gradient of $f_0(\cdot;y)$, we conclude for all x

$$\begin{aligned} f_\kappa(y^0;y) = f_0(y^0;y) + \psi(y^0) &\leq f_0(y;y) + \nabla f_0(y;y)^T(y_0-y) + \frac{\kappa+L}{2}\|y_0-y\|^2 + \psi(y_0) \\ &\leq f_0(y;y) + \nabla f_0(y;y)^T(x-y) + \frac{\kappa+L}{2}\|x-y\|^2 + \psi(x). \end{aligned} \tag{D.36}$$

By setting $x = y^*$ in both (D.35) and (D.36) and combining these results, we conclude

$$f_\kappa(y^0;y) \leq f_\kappa^*(y) + \frac{\kappa+L}{2}\|y^*-y\|^2.$$

□

Note that if we are not in the composite setting and $\kappa > \rho$, then $f_\kappa(\cdot,y)$ is $(\kappa+L)$-strongly convex. Using standard bounds for strongly convex functions, Equation (D.34) follows (see [Nesterov, 2004]). We next show an important lemma for deducing the inner complexities.

Lemma 26. *Assume $\kappa > \rho$. Given any $\varepsilon \leq \frac{\kappa-\rho}{2}$, if an iterate z satisfies dist$(0, \partial f_\kappa(z;y)) \leq \varepsilon\|y^*-y\|$, then*

$$\text{dist}(0, \partial f_\kappa(z;y)) \leq 2\varepsilon\|z-y\|. \tag{D.37}$$

Proof. Since $\kappa > \rho$, we know $f_\kappa(\cdot; y)$ is $(\kappa - \rho)$-strongly convex. Therefore, by [Nesterov, 2004], we know

$$\|z - y^*\| \leq \frac{1}{\kappa - \rho} \text{dist}(0, \partial f_\kappa(z; y)). \tag{D.38}$$

By the triangle inequality and Equation (D.38), we deduce

$$\begin{aligned}
\text{dist}(0, \partial f_\kappa(z; y)) \leq \varepsilon \|y^* - y\| &\leq \varepsilon \Big(\|y^* - z\| + \|z - y\| \Big) \\
&\leq \frac{\varepsilon}{\kappa - \rho} \cdot \text{dist}(0, \partial f_\kappa(z; y)) + \varepsilon \|z - y\| \\
&\leq \frac{1}{2} \cdot \text{dist}(0, \partial f_\kappa(z; y)) + \varepsilon \|z - y\| .
\end{aligned}$$

The last inequality follows because of the assumption $\varepsilon \leq \frac{\kappa - \rho}{2}$. Rearranging the terms above, we get the desired result. □

These two lemmas together give us Theorem 14.

Proof of Theorem 14. First, we prove that z_T satisfies both adaptive stationary condition and the descent condition. Recall, the point y^0 is defined to be the prox or y depending on if $f_\kappa(\cdot; y)$ is a composite form or smooth, respectively (see statement of Theorem 14). By Lemma 25 (or the remark following it), the starting y^0 satisfies

$$f_\kappa(y^0; y) - f_\kappa^*(y) \leq \frac{\kappa + L}{2} \|y^* - y\|^2 .$$

By the linear convergence assumption of $\mathcal{M}$ (see (D.14)) and the above equation, after $T := T_\kappa$ iterations initializing from y^0, we have

$$\begin{aligned}
\text{dist}^2(0, \partial f_\kappa(z_T; y)) &\leq A_\kappa (1 - \tau_\kappa)^T \Big(f_\kappa(y^0; y) - f_\kappa^*(y) \Big) \\
&\leq A_\kappa e^{-T \cdot \tau_\kappa} \Big(f_\kappa(y^0; y) - f_\kappa^*(y) \Big) \\
&\leq \frac{(\kappa - \rho)^2}{32(L + \kappa)} \cdot \frac{L + \kappa}{2} \|y^* - y\|^2 \\
&\leq \frac{(\kappa - \rho)^2}{64} \|y^* - y\|^2 .
\end{aligned} \tag{D.39}$$

By applying Lemma 26, we obtain

$$\text{dist}(0, \partial f_\kappa(z_T; y)) \leq \frac{(\kappa - \rho)}{4} \|z_T - y\| \leq \kappa \|z_T - y\| ,$$

which gives the adaptive stationary condition. Next, we show we also have the descent condition. By the $(\kappa - \rho)$-strong convexity of $f_\kappa(\cdot; y)$, for any $v \in \partial f_\kappa(z_T; y)$ we deduce

$$\begin{aligned} f_\kappa(y; y) &\geq f_\kappa(z_T; y) + \langle v, y - z_T \rangle + \frac{\kappa - \rho}{2} \|z_T - y\|^2 \\ &\geq f_\kappa(z_T; y) - \|v\| \|y - z_T\| + \frac{\kappa - \rho}{2} \|z_T - y\|^2 \\ &\geq f_\kappa(z_T; y). \end{aligned}$$

This yields the descent condition which completes the proof for T. The proof for S_κ is similar to T_κ, so we omit many of the details. In this case, we only need to show the adaptive stationary condition. For convenience, we denote $S = S_\kappa$. Following the same argument as in Equation (D.39) but with $S \log(k+1)$ number of iterations, we deduce

$$\operatorname{dist}^2(0, \partial f_\kappa(z_S; y)) \leq \frac{(\kappa - \rho)^2}{64(k+1)^2} \|y^* - y\|^2 .$$

By applying Lemma 26, we obtain

$$\operatorname{dist}(0, \partial f_\kappa(z_S; y)) \leq \frac{(\kappa - \rho)}{4} \|z_T - y\| \leq \frac{\kappa}{k+1} \|z_S - y\| ,$$

which proves the desired result for z_S. □

Assuming Proposition 21 and Proposition 22 hold as well as Lemma 27, we begin by providing the proof of Theorem 16.

Proof of Theorem 16. We consider two cases: (i) the function f is non-convex and (ii) the function f is convex. First, we consider the non-convex setting. To produce $\bar{x}_k$, the method $\mathcal{M}$ is called

$$T \log\left(\tfrac{4L}{\kappa_0}\right) / \log(2) \tag{D.40}$$

number of times. This follows from Proposition 21 and Lemma 27. The reasoning is that once $\kappa > \rho + L$, which only takes at most $\log(4L/\kappa_0)$ number of increases of κ to reach, then the iterate $\bar{x}_k$ satisfies the stopping criteria (D.18). Each time we increase κ we run $\mathcal{M}$ for T iterations. Therefore, the total number of iterations of $\mathcal{M}$ is given by multiplying T with $\log(4L/\kappa_0)$. To produce $\tilde{x}_k$, the method $\mathcal{M}$ is called $S \log(k+1)$ number of times. (Note: the proof of Theorem 15 does not need $\tilde{x}_k$ to satisfy (D.19) in the non-convex case).

Next, suppose the function f is convex. As before, to produce $\bar{x}_k$ the method $\mathcal{M}$ is called (D.40) times. To produce $\tilde{x}_k$, the method $\mathcal{M}$ is called $S \log(k+1)$ number of times. By Proposition 22, the iterate $\tilde{x}_k$ satisfies (D.19); a key ingredient in the proof of Theorem 15. □

D.10.1 Inner complexity for $\bar{x}_k$: proof of Proposition 21

Next, we supply the proof of Proposition 21 which shows that by choosing κ large enough, Algorithm 11 terminates.

Proof of Proposition 21. The idea is to apply Theorem 14. Since the parameter A_κ increases with κ, then we upper bound it by $A_{\kappa_k} \leq A_{4L}$. Moreover, we have $\kappa - \rho \geq \rho + L - \rho = L$. Lastly, since τ_κ is increasing in κ, we know $\frac{1}{\tau_\kappa} \leq \frac{1}{\tau_L}$. Plugging these bound into Theorem 14, we see that for any smoothing parameter κ satisfying $\rho + L < \kappa < 4L$, we get the desired result. □

Next, we compute the maximum number of times we must double κ until $\kappa > \rho + 2L$.

Lemma 27 (Doubling κ)**.** *If we set T and S according to Theorem 16, then the doubling of κ_0 will terminate as soon as $\kappa > \rho + L$. Thus the number of times κ_0 must be doubled in Algorithm 11 is at most*

$$\frac{\log\left(\frac{2(\rho+L)}{\kappa_0}\right)}{\log(2)} \leq \left\lceil \frac{\log\left(\frac{4L}{\kappa_0}\right)}{\log(2)} \right\rceil.$$

Since κ is doubled (Algorithm 11) and T is chosen as in Proposition 21 , the maximum the value κ, $\kappa_{\max}$, takes is $2(\rho + L) \leq 4L$.

D.10.2 Inner complexity for $\tilde{x}_k$: proof of Proposition 22

In this section, we prove Proposition 22, an inner complexity result for the iterates $\tilde{x}_k$. Recall that the inner-complexity analysis for $\tilde{x}_k$ is important only when f is convex (see Section D.4). Therefore, we assume throughout this section that the function f is convex. We are now ready to prove Proposition 22.

Proof of Proposition 22. The proof immediately follows from Theorem 14 by setting $\kappa = \kappa_{\text{cvx}}$ and $\rho = 0$ as the function f is convex. □

Bibliography

A. Agarwal and L. Bottou. A lower bound for the optimization of finite sums. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2015.

N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, and T. Ma. Finding approximate local minima for nonconvex optimization in linear time. *arXiv:1611.01146*, 2016.

Z. Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv:1603.05953*, 2016.

Z. Allen-Zhu. Natasha: Faster stochastic non-convex optimization via strongly non-convex parameter. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2017.

Z. Allen-Zhu and L. Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv:1407.1537*, 2014.

Y. Arjevani and O. Shamir. Dimension-free iteration complexity of finite sum optimization problems. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

A. Auslender. Numerical methods for nondifferentiable convex optimization. *Nonlinear Analysis and Optimization*, pages 102–126, 1987.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

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.

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

D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

D. P. Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, 129(2):163–195, 2011.

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

A. Bietti and J. Mairal. Stochastic optimization with variance reduction for infinite datasets with finite-sum structure. *arXiv:1610.00970*, 2016.

J. Bolte, T.P. Nguyen, J. Peypouquet, and B. Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Mathematical Programming, Serie A*, 2016.

J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. A family of variable metric proximal methods. *Mathematical Programming*, 68(1-3):15–47, 1995.

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

J. M. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media, 2010.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT*. Springer, 2010.

L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2009.

C. G. Broyden. Quasi-newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.

C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

S. Bubeck, Y. Lee, and M. Singh. A geometric alternative to nesterov's accelerated gradient descent. *arXiv:1506.08187*, 2015.

J. V. Burke and M. Qian. A variable metric proximal point algorithm for monotone operators. *SIAM Journal on Control and Optimization*, 37(2):353–375, 1999.

J. V. 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.

R. H. Byrd and J. Nocedal. A tool for the analysis of quasi-newton methods with application to unconstrained minimization. *SIAM Journal on Numerical Analysis*, 26(3):727–739, 1989.

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

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.

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

Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Accelerated methods for non-convex optimization. *arXiv:1611.00756*, 2016.

Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower Bounds for Finding Stationary Points I. *preprint arXiv:1710.11606*, 2017.

Y. Carmon, O. Hinder, J. C. Duchi, and A. Sidford. "convex until proven guilt": Dimension-free acceleration of gradient descent on non-convex functions. *preprint arXiv:1705.02766*, 2017.

C. Cartis, N. I. M. Gould, and P. L. Toint. On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6):2833–2852, 2010.

C. Cartis, N. I. M. Gould, and P. L. Toint. On the complexity of finding first-order critical points in constrained nonlinear optimization. *Mathematical Programming*, 2014.

A. Chambolle and T. Pock. A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions. *SMAI Journal of Computational Mathematics*, 1:29–54, 2015.

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

F. H. Clarke, R. J. Stern, and P. R. Wolenski. Proximal smoothness and the lower-C^2 property. *Journal of Convex Analysis*, 2(1-2):117–144, 1995.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

R. Correa and C. Lemaréchal. Convergence of some algorithms for convex minimization. *Mathematical Programming*, 62(1):261–275, 1993.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57 (11), 2004.

W. C. Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1 (1):1–17, 1991.

Etienne De Klerk, François Glineur, and Adrien B Taylor. On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions. *Optimization Letters*, 11(7):1185–1199, 2017.

A. J. Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

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

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

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.

J. E. Dennis and J. J. Moré. Quasi-newton methods, motivation and theory. *SIAM review*, 19 (1):46–89, 1977.

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

D. Drusvyatskiy and C. Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Preprint arXiv:1605.00125*, 2016.

D. Drusvyatskiy, M. Fazel, and S. Roy. An optimal first order method based on optimal quadratic averaging. *arXiv:1604.06543*, 2016.

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

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

H. Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93: 418–491, 1959.

O. Fercoq and Z. Qu. Restarting accelerated gradient methods with a rough strong convexity estimate. *arXiv:1609.07358*, 2016.

R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3): 317–322, 1970.

R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *The computer journal*, 6(2):163–168, 1963.

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

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer series in statistics New York, 2001.

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

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

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

D. Gabay. Chapter ix applications of the method of multipliers to variational inequalities. *Studies in mathematics and its applications*, 15:299–331, 1983.

R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points-online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, 2015.

S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2, Ser. A):59–99, 2016.

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

P. Giselsson and M. Fält. Nonsmooth minimization using smooth envelope functions. *preprint arXiv:1606.01327*, 2016.

D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

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

O. Güler. On the convergence of the proximal point algorithm for convex minimization. *SIAM Journal on Control and Optimization*, 29(2):403–419, 1991.

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

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning With Sparsity: The Lasso And Generalizations*. CRC Press, 2015.

B. He and X. Yuan. An accelerated inexact proximal point algorithm for convex minimization. *Journal of Optimization Theory and Applications*, 154(2):536–548, 2012.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms I*. Springer, 1996.

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

L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009.

R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12(Oct):2777–2824, 2011.

C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. *arXiv:1703.00887*, 2017.

C. Jin, P. Netrapalli, and M. I. Jordan. Accelerated Gradient Descent Escapes Saddle Points Faster than Gradient Descent. *preprint arXiv:1711.10456*, 2017.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

G. Lan and Y. Zhou. An optimal randomized incremental gradient method. *arXiv:1507.02000*, 2015.

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

J. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent only converges to minimizers. In *Conference on Learning Theory*, pages 1246–1257, 2016.

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

C. Lemaréchal and C. Sagastizábal. Variable metric bundle methods: from conceptual to implementable forms. *Mathematical Programming*, 76(3):393–410, 1997b.

H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

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

H. Lin, J. Mairal, and Z. Harchaoui. A generic quasi-newton algorithm for faster gradient-based optimization. *arXiv:1610.00960v2*, 2017.

Q. Lin, Z. Lu, and L. Xiao. An accelerated proximal coordinate gradient method. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

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

J. Mairal. Optimization with first-order surrogate functions. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

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

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.

S. Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

B. Martinet. Régularisation d'inéquations variationnelles par approximations successives. *Revue française d'informatique et de recherche opérationnelle, série rouge*, 4(3):154–158, 1970.

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

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

J. J. Moré and J. A. Trangenstein. On the global convergence of broyden's method. *Mathematics of Computation*, pages 523–540, 1976.

J. J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes Rendus de l'Académie des Sciences de Paris*, 255:2897–2899, 1962.

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

T. Murata and T. Suzuki. Stochastic dual averaging methods using variance reduction techniques for regularized empirical risk minimization problems. *arXiv:1603.02412*, 2016.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

A. Nemirovskii, D. B. Yudin, and E. R. Dawson. *Problem complexity and method efficiency in optimization*. Wiley, 1983.

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.

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

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

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Y. Nesterov. How to make the gradients small. *OPTIMA, MPS Newsletter*, pages 10–11, 2012a.

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

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

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

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

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

B. O'donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

M. O'Neill and S. J. Wright. Behavior of Accelerated Gradient Methods Near Critical Points of Nonconvex Problems. *ArXiv e-prints*, June 2017.

C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui. Catalyst acceleration for gradient-based non-convex optimization. *arXiv preprint arXiv:1703.10993v2*, 2017.

N. Parikh and S. P. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1 (3):123–231, 2014.

R. A. Poliquin and R. T. Rockafellar. Prox-regular functions in variational analysis. *Transactions of the American Mathematical Society*, 348(5):1805–1838, 1996.

M. J. D. Powell. A new algorithm for unconstrained optimization. *Nonlinear programming*, pages 31–65, 1970.

M. J. D. Powell. On the convergence of the variable metric algorithm. *IMA Journal of Applied Mathematics*, 7(1):21–36, 1971.

M. J. D. Powell. How bad are the BFGS and DFP methods when the objective function is quadratic? *Mathematical Programming*, 34(1):34–47, 1986.

H. Raguet, J. Fadili, and G. Peyré. A generalized forward-backward splitting. *SIAM Journal on Imaging Sciences*, 6(3):1199–1226, 2013.

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.

S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning (ICML)*, 2016a.

S. J. Reddi, S. Sra, B. Poczos, and A. J. Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2016b.

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.

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

R. T. Rockafellar. Favorable classes of Lipschitz-continuous functions in subgradient optimization. In *Progress in nondifferentiable optimization*, volume 8 of *IIASA Collaborative Proc. Ser. CP-82*, pages 125–143. Internat. Inst. Appl. Systems Anal., Laxenburg, 1982.

R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1998.

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

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

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

M. Schmidt, N. Le Roux, and F. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2011b.

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

D. Scieur, V. Roulet, F. Bach, and A. d'Aspremont. Integration methods and accelerated optimization algorithms. *arXiv:1702.06751*, 2017.

S. Shalev-Shwartz. SDCA without duality, regularization, and individual convexity. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2016.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

S. Shalev-Shwartz and T. Zhang. Proximal stochastic dual coordinate ascent. *arXiv:1211.2717*, 2012.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

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

D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

N. Z. Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

M. V. Solodov and B. F. Svaiter. A unified framework for some inexact proximal point algorithms. *Numerical Functional Analysis and Optimization*, 2001.

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.

Adrien B Taylor, Julien M Hendrickx, and François Glineur. Exact worst-case performance of first-order methods for composite convex optimization. *SIAM Journal on Optimization*, 27 (3):1283–1313, 2017.

A. Themelis, L. Stella, and P. Patrinos. Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone line-search algorithms. *preprint arXiv:1606.06256*, 2016.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 01 2008.

Y. Z. Tsypkin. *Foundations of the theory of learning systems*. Academic Press New York, 1973.

Y. Z. Tsypkin and Z. J. Nikolic. *Adaptation and learning in automatic systems*, volume 73. Academic Press New York, 1971.

V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

J. H. Wilkinson. *The algebraic eigenvalue problem*, volume 87. Clarendon Press Oxford, 1965.

B. E. Woodworth and N. Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

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

K. Yosida. Functional analysis. *Berlin-Heidelberg*, 1980.

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

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Y. Zhang and L. Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015a.

Y. Zhang and L. Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of the International Conferences on Machine Learning (ICML)*, 2015b.

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.