Contributions to Federated Learning and First-Order Optimization

Aymeric Dieuleveut

HABILITATION À DIRIGER DES RECHERCHES

Discipline : Mathématiques

Spécialité : Statistiques et Optimisation

présentée par

Aymeric DIEULEVEUT

Contributions to Federated Learning and first order optimization.

Soutenue le 28 Mars 2023

Dan ALISTARH,	IST Austria	Rapporteur
lexandre D'ASPREMONT	ENS, Inria	Examinateur
Émilie CHOUZENOUX	Inria, CENTRALESUPELEC	Examinatrice
Sébastien GADAT	TSE	Examinateur
Vianney PERCHET	ENSAE	Rapporteur
Michael I. JORDAN	UC Berkeley	Rapporteur

Centre de Mathématiques Appliquées (CMAP)
Ecole Polytechnique
Institut Polytechnique de Paris
Route de Saclay,
91120, Palaiseau

Anything that happens, happens.

Anything that, in happening, causes something else to happen, causes something else to happen.

Anything that, in happening, causes itself to happen again, happens again.

It doesn't necessarily do it in chronological order, though.

Douglas Adams,

Mostly Harmless

1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
75025
121393
196418
317811
514229
832040
1346269
2178309
3524578
5702887
9227465
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437

The first occurrence of the number 43 in the Fibonacci sequence is 433494437.
Apart from 43 (twice + once reversed), the remaining digits, are 4,9 and 7.
Observe that $4 \times 9 + 7 = 43$.
One could call that a coincidence.

This number is u_{43}. Maybe it's destiny.

Personal discovery, 2007

to those from whom I learned

&

to the fools who dream

Remerciements

Je tiens à remercier toutes celles et ceux qui ont accompagné mon parcours personnel et de recherche au cours de ces dernières années.

Pour commencer, les rapporteurs de ce manuscrit et le jury. In particular, Dan Alistarh, Michael Jordan, and Vianney Perchet, for their detailed reports and kind comments. I hope we have the chance to work together in the future. Un merci supplémentaire pour Vianney, ta vitesse d'exécution a été une aide précieuse pour naviguer dans le processus inextricable des HDR saclaysiennes, tu as été exceptionnel.

Quant au jury, Alexandre d'Aspremont, Émilie Chouzenoux, Sébastien Gadat, merci pour votre disponibilité. Vous compter parmi les membres du jury est un plaisir et un honneur.

Ma seconde pensée va à mes collaborateurs et mentors, sans lesquels ce travail n'aurait pas été possible. En tout premier lieu, Francis, *again*, car quand on sort de thèse on est encore bien naïf et on mesure après seulement tout ce que tu fais pour ton labo, la communauté, et tes (ex-)étudiants. Tu es un modèle par beaucoup d'aspects. Martin, le postdoc à Lausanne a été un excellent moment et je te remercie de m'y avoir s'y bien accueilli.

Ensuite, of course, ceux qui m'ont fait venir à l'X, et avec qui j'ai tant travaillé depuis, en particulier Éric, Julie, Erwan S. Éric, quand tu m'as appelé il y a 5 ans maintenant, pour me parler de ce poste à l'X, je ne mesurais pas l'impact que cet appel aurait :) Merci pour ta remarquable motivation et ton exigence. J'espère avoir autant d'énergie que toi dans ma recherche future. Merci bien sûr également pour le soutien, c'est toi qui m'as permis de recruter mes premiers doctorants. Julie, merci pour tes précieux conseils, c'est aussi grâce à toi que mon séjour à l'X s'est passé au mieux. Encadrer une thèse avec toi est un grand plaisir, et ton équilibre de vie fait rêver. Erwan, c'est une joie systématique de travailler avec toi, en recherche comme en enseignement. Merci pour ta bonne humeur, ton calme et ton recul. Tes conseils, du badminton à la recherche, sont toujours bienvenus et d'une sagesse impénétrable.

Mes autres collaborateurs ensuite, qui font du métier de chercheur ce qu'il a de merveilleux. Gersende, merci pour ta rigueur et tout ce que tu as fait pour les projets, tant coté enseignement que recherche. Adrien, merci ton enthousiasme communicatif, ta passion, et tout ce que tu nous as appris. Claire, merci pour ta vision de la recherche, ton sérieux, et bien sûr aussi les moments de craquage partagés avec Erwan et toi.

Yannig, Olivier, Alain, Aurélien, Nicolas, Édouard, Damien, Fabian, Gilles, Geneviève, travailler avec vous sur les divers projets a été passionnant, et j'espère recommencer! Thank you Yaniv, working with you is great too.

Aurais-je dû commencer par vous? Constantin, Baptiste, Margaux, Alexis, vous avoir eu comme mes premiers thésards a été une de mes plus grandes joies de ces dernières années. Vous êtes exceptionnels :) Constantin, par ton enthousiasme et ta volonté d'apprendre.

Baptiste, par ta soif de mathématiques, les digressions ou discussions interminables sur les clôtures algébriques à des heures parfois tardives. Margaux, par ton intégrité et ta pugnacité, ton exigence et ton enthousiasme scientifique et au-delà. Alexis, par ta persévérance et ta bonne humeur. Votre aide et votre disponibilité quand j'en avais besoin ont aussi été irréprochables.

Je n'oublie pas les autres étudiants qui ont participé à certains projets, en particulier, Scott, Maxence, Céline, Vincent, Maxime, Louis, Kumar.

Au-delà des collaborateurs de recherche, vous êtes nombreux à faire de l'X un lieu qui donne envie d'y rester. En tout premier lieu, Marylou, pour les footings, les cafés, les partages, ta motivation, ta vision équilibrée et ton efficacité. J'ai bien de la chance de partager ton bureau :)

Ensuite, toutes les personnes qui font tourner la grande machine, à tous les niveaux. Emmanuel, Josselin, merci pour votre confiance quant à l'enseignement (et dans le reste), vous n'avez pas hésité à me donner la responsabilité de cours que j'adore enseigner. Grégoire, Aline, et Thierry merci pour votre travail pour le labo. Merci aussi à Nassera, Alexandra, Alex, Marine, Nathalie, Stéphanie, Aldjia votre soutien et travail pour les labos et départements sont infaillibles. Merci aussi à Mahdi, Rémi, Anna, Karim, Erwan LP, Clément, Igor, Stéfano, Mathieu L, et tous mes collègues d'enseignement, de département ou de laboratoire ! A Hadrien avec qui j'ai hâte de travailler. Merci aussi aux jeunes de l'équipe, et à ceux qui vont bientôt débuter (Rémi, Mahmoud, Renaud, Damien)! En plus de tous ceux que j'ai cités plus haut, je pense à Guillaume C, Pierre C., Solange, Louis, Gabriel, Manon, Clément M., Clément G., Wassim, Antoine, votre participation à l'animation de l'équipe est cruciale, et je sais que votre solidarité est précieuse.

Enfin, j'ai une pensée particulière pour tous les étudiants et étudiantes des cursus dans lesquels j'ai la chance d'enseigner. Votre curiosité et votre dynamisme sont les sources de mon bonheur à enseigner.

Les derniers mots de ces remerciements vont bien-sûr à mes amis et mes proches. Je pense à Vincent V., Anne C., Yana, Lénaïc, Evann, Raphaël, Loucas, Pierre A., Thomas M., Iryna.

Enfin, j'ai une pensée pour ma famille. Pour Papa bien-sûr, même si tu n'es plus là depuis janvier. Pour Nicolas, Rémi, et surtout Floriane, Anouk, Daphné, Maman, vous êtes fantastiques. Jordane bien entendu, ton soutien et ta patience au quotidien sont la clé de tout.

Abstract

The design of new algorithms for artificial intelligence is one of the major challenges of our time. The extraordinary progress made in recent years has been enabled by a combination of factors: on the one hand, the simultaneous growth of available data sets and computing power; on the other hand, algorithmic innovations that have played a decisive role in making it possible to move to the next scale. During the last years, the field of *Federated Learning* (FL) has gained a lot of importance.

In Federated Learning, several organizations or devices seek to collaboratively train a model under the orchestration of a central server, while keeping individual datasets on their respective local storage. Federated Learning has emerged as a fundamental setting to tackle new societal and industrial challenges in machine learning. Indeed, privacy has become a major concern for both society (individuals participating into the training want to protect their privacy) and industries (facing legal constraints preventing them from exploiting valuable data). Overall, it becomes necessary to train the models without centralizing the data, either because of those privacy constraints, or sometimes because extremely large datasets have to be distributed over networks of storing devices. Consequently, new optimization challenges are arising, taking into account communication constraints, and new opportunities to improve the models to adapt to the users are being explored.

Developing new algorithms for Federated Learning is thus a key challenge. It will simultaneously positively impact society, by protecting individual data and restoring public trust and confidence in machine learning technologies, and unlock countless novel opportunities of collaboration between entities willing to collaborate without centralizing their datasets; a crucial situation in medical applications, fraud detection, IOT, and many other domains.

The design and analysis of algorithms and techniques for optimization and for optimization in FL have been at the core of my research interests over the last few years. Designing new techniques requires an in-depth understanding of both the (first-order) optimization techniques and the specificities of the federated context.

In this manuscript, I summarize my research activities over the last years and provided a detailed description of some featured contributions, especially in the domain of Federated Learning with communication constraints, in the first part of the manuscript, and fundamental results on classical first-order optimization, in the second part.

Keywords: stochastic approximation, first-order optimization, federated learning, compression, performance estimation.

Résumé

La conception de nouveaux algorithmes pour l'intelligence artificielle est l'un des défis majeurs de notre époque. Les progrès extraordinaires réalisés ces dernières années ont été rendus possibles par une combinaison de facteurs : d'une part, la croissance simultanée des ensembles de données disponibles et de la puissance de calcul ; d'autre part, les innovations algorithmiques qui ont joué un rôle décisif pour permettre le passage à l'échelle supérieure. Au cours des dernières années, le domaine de l'apprentissage fédéré (FL) a pris beaucoup d'importance.

Dans le cadre de l'apprentissage fédéré, plusieurs organisations ou dispositifs cherchent à former un modèle en collaboration sous l'orchestration d'un serveur central, tout en conservant les ensembles de données individuels sur leur stockage local respectif. L'apprentissage fédéré est apparu comme un cadre fondamental pour relever les nouveaux défis sociétaux et industriels de l'apprentissage automatique. En effet, le respect de la vie privée est devenu une préoccupation majeure tant pour la société (les individus participant à l'apprentissage veulent protéger leur vie privée) que pour les industries (confrontées à des contraintes légales les empêchant d'exploiter des données précieuses). Globalement, il devient nécessaire d'entraîner les modèles sans centraliser les données, soit en raison de ces contraintes de confidentialité, soit parfois parce que des ensembles de données extrêmement volumineux doivent être distribués sur des réseaux de dispositifs de stockage. Par conséquent, de nouveaux défis d'optimisation apparaissent, en tenant compte des contraintes de communication, et de nouvelles possibilités d'améliorer les modèles pour les adapter aux utilisateurs sont explorées.

Le développement de nouveaux algorithmes pour l'apprentissage fédéré constitue donc un défi majeur. Il aura simultanément un impact positif sur la société, en protégeant les données individuelles et en rétablissant la confiance du public dans les technologies d'apprentissage automatique, et débloquera d'innombrables opportunités inédites de collaboration entre des entités désireuses de collaborer sans centraliser leurs ensembles de données ; une situation cruciale dans les applications médicales, la détection des fraudes, l'IOT et bien d'autres domaines.

La conception et l'analyse d'algorithmes et de techniques d'optimisation et d'optimisation en FL ont été au coeur de mes intérêts de recherche au cours des dernières années. La conception de nouvelles techniques nécessite une compréhension approfondie à la fois des techniques d'optimisation (de premier ordre) et des spécificités du contexte fédéré.

Dans ce manuscrit, je résume mes activités de recherche au cours des dernières années et j'ai fourni une description détaillée de certaines contributions marquantes,

notamment dans le domaine de l'apprentissage fédéré avec contraintes de communication, dans la première partie du manuscrit, et des résultats fondamentaux sur l'optimisation classique du premier ordre, dans la seconde partie.

Contents

Contributions and manuscript outline

Research directions and featured contributions.

Over the last few years, my research interests have been centered around several aspects of optimization for machine learning. Building on the work done during my PhD thesis, centered on stochastic algorithms for high-dimensional learning, I have kept an active line of research on stochastic optimization, with a significant focus on the distributed and federated frameworks, in which multiple agents participate in the optimization process. Simultaneously, I also worked on "classical" optimization, to provide guarantees in the context of centralized optimization.

Some results from these two research directions are highlighted in this manuscript, organized in two main parts. In the first part of the manuscript, the first chapter provides a general summary of the framework and the results. The subsequent chapters give more detailed results of some selected contributions.

Beyond these contributions, I have also maintained several other lines of work, especially on prediction with missing data, machine learning for time series and uncertainty quantification with conformal prediction.

Part I: Contributions to Federated Learning.

In the first part of the manuscript, I highlight results on learning algorithms and methods for federated learning with communications constraints.

In Chapter 2, we describe how control variates can be used to alleviate the impact of heterogeneity in FL methods relying on compression. We provide convergence results for both the convex framework, with bi-directional compression [P1], and non-convex frameworks in the case of the EM algorithm [C6]. Similar results were also obtained for Langevin algorithms [C8] (this latest example is not featured in the manuscript).

In Chapter 3, we show how *model preservation*, which relies on compensating the errors made by the downlink compression in a federated scheme can be used to recover nearly optimal convergence rate in the context of bi-directional compression [C7].

My line of work on the federated framework was initiated at the end of my postdoc at EPFL, and driven by my work with my first PhD student Constantin Philippenko, who I have supervised since December 2019 and is expected to defend in Spring 2023. The other papers featured in the manuscript correspond to other collaborations.

Several other contributions, that are not highlighted in the manuscript, were made on that topic. This includes convergence analysis for homogeneous distributed stochastic algorithms with local updates [C1], differential privacy with heterogeneity [C10], a benchmarking suite with a consortium of collaborators [C13], as well as multiple ongoing works.

Part II: Contributions to first-order optimization

The second part of the manuscript focuses on contributions made on *centralized* optimization. Relying on an in-depth understanding of optimization algorithms is crucial to adapt them to distributed frameworks. In this part, I chose to highlight recent contributions on deterministic first order optimization. An important part of our work in that direction relies on computer assisted proofs for optimization, especially with performance estimation programs. Performance estimation programs enable the numerical derivation of convergence rates for large classes of algorithms.

In Chapter 4, I describe PEPit, a Python package for performance estimation developed with Baptiste Goujaud (who started his PhD under my supervision in October 2020) and collaborators [P2]. PEPit enables the automatic derivation of numerical worst-case analyses of a large family of first-orderoptimization methods possibly involving gradient, projection, proximal, or linear optimization oracles, along with their approximate, or Bregman variants. We leveraged this framework to obtain optimal convergence rates for non-smooth functions that have a quadratic growth around the optimal point [P4].

In Chapter 5, I present several results on the particular case of quadratic optimization, a simple yet powerful framework to provide refined analysis of algotihms. First, in [W1], a joint work with Baptiste and Adrien Taylor we show how to incorporate momentum in Polyak step-size algorithm an open question that admits a remarkably simple answer in that framework, linked to conjugate gradient algorithms. Then we show how a faster acceleration than the one achieved by Polyak momentum can be achieved on quadratic optimization that have clustered eignevalues. This is obtained by using cyclical step-sizes, an idea that was motivated by techniques widely used in deep-learning [C9].

Results on stochastic methods, including convergence diagnostics for constant step size in SGD [C4], that follows our refined analysis of constant learning rate SGD [J3]; as well as results from my PhD thesis on least squares regression and in the infinite dimensional regime [J1, J2] are not featured in the manuscript.

Other contributions

1. **Times series and applications.** I became more interested in time series during my postdoc at EPFL. With Jean-Yves Franceschi, whose internship at EPFL I supervised, and Martin Jaggi, we proposed a novel approach to obtain unsupervised representations of times series [C2], based on a triplet loss and causal convolutional neural networks. This work, published in Neurips 2019, was very well received by the community. It was successfully reproduced in the Neurips reproducibility challenge, and has been widely re-used (300+ stars, 80+ forks on github).

   Working on time series was a strong motivation to supervise Margaux Zaffran's CIFRE PhD thesis with EDF (French Electricity group) on the prediction of spot price on electricity markets, started in December 2020. Our work, focused on uncertainty prediction for time series with conformal methods [C11], was published at ICML2022, and introduces a novel algorithm for conformal prediction for time series.

2. **Prediction with missing data.** Finally, my the last notable and active line of work is the theoretical study of prediction with missing data. In [C5] we provided fast convergence rates for stochastic gradient descent for least squares regression with missing data. With Erwan Scornet and Claire Boyer, we supervise the PhD of Alexis

Ayme, who started in September 2021. His first article [C12], published at (ICML 2022) provides a minimax rate for prediction with missing data.

Supervision

1. PhD supervision
    - Constantin Philippenko, started December 2019, supervised with Eric Moulines.
    - Baptiste Goujaud, started October 2020, supervised with Eric Moulines.
    - Margaux Zaffran, started December 2020, supervised with Julie Josse, Olivier Féron, Yannig Goude.
    - Alexis Ayme, started September 2021, supervised with Claire Boyer and Erwan Scornet.
2. Internships supervision
    - Scott Pesme, spring-summer 2019, at EPFL (with Nicolas Flammarion), now PhD student with N. Flammarion, EPFL.
    - Jean-Yves Franceschi, spring-summer 2019, at EPFL (with Martin Jaggi), finished his PhD in 2021, now researcher at Criteo
    - Maxence Noble, spring-summer 2021, at Polytechnique (with Aurélien Bellet), now PhD student with A. Durmus, Polytechnique.
    - K.K. Patel, spring summer 2018, now PhD student with N. Srebro, TTIC.

List of publications and preprints

Alias: J - Journal, C - International conference, W - International workshop, P - Preprint. All preprints are currently under review.

On Federated Learning and connected topics:

Reference	Alias	Topic	Conference or Journal
Terrail et al. [124]	[C13]	Benchmarking suite for FL	Neurips 2022
Noble et al. [86]	[C10]	Differential privacy and Federated Learning with heterogeneity	Aistats 2022
Leconte et al. [68]	[P5]	Gaussian randomized vector compression for FL	Preprint 2022
Vono et al. [127]	[C8]	Quantized Langevin algorithm in FL	Aistats 2022
Dieuleveut et al. [25]	[C6]	Federated EM algorithm with compression	Neurips 2021
Philippenko and Dieuleveut [96]	[C7]	Preserved model for Federated Learning with bi-directional compression	Neurips 2021
Philippenko and Dieuleveut [95]	[P1]	Federated learning with bi-directional compression and heterogeneity	Preprint 2020
Dieuleveut and Patel [22]	[C1]	Convergence of Local SGD	Neurips 2019

On convergence guarantees for first order optimization, in a centralized framework:

Reference	Alias	Topic	Conference or Journal
Goujaud et al. [44]	[P8]	Discovering cycles with PEP to build counter examples	Preprint 2023
Dieuleveut et al. [26]	[P7]	Survey on Stochastic Approximation for signal processing and ML	Preprint 2023
Goujaud et al. [42]	[P4]	Optimal convergence rates for non smooth optimization with quadratic growth	Preprint 2022
Goujaud et al. [43]	[W1]	Polyak-step size with momentum	NeuripsOPT 2022
Goujaud et al. [40]	[P2]	Performance Estimation for first order optimization	Preprint 2022
Goujaud et al. [41]	[C9]	Faster acceleration for quadratic optimization with spectrum constraints	Aistats 2022
Pesme et al. [94]	[C4]	Convergence diagnostic for SGD	ICML 2022
Dieuleveut et al. [24]	[J3]	Improved analysis of SGD with constant learning rate	Annals of statistics, 2020
Dieuleveut et al. [23]	[J2]	Optimal bias and variance for accelerated least squares regression	JMLR 2017
Dieuleveut and Bach [21]	[J1]	Optimal statistical rate for stochastic approximation in Hilbert spaces	Annals of Statistics, 2016

On prediction with missing data:

Reference	Alias	Topic	Conference or Journal
Zaffran et al. [133]	[P7]	Conformal prediction with missing data	Preprint 2023
Ayme et al. [8]	[P6]	Implicit regularization missing data	Preprint 2023
Ayme et al. [7]	[C12]	Minimax rates for prediction with missing data	ICML 2022
Sportisse et al. [111]	[C5]	Stochastic gradient descent with missing data	Neurips 2020

On Learning representations, times series and uncertainty quantification:

Reference	Alias	Topic	Conference or Journal
Zaffran et al. [134]	[C11]	Conformal prediction for time series	ICML 2022
Singh et al. [109]	[C3]	Optimal transport for text representations	Aistats 2020
Franceschi et al. [32]	[C2]	Unsupervised representation for times series	Neurips 2019

Part I

Federated Learning, stochastic algorithms and compression

1

General introduction to Federated Learning - summary of the main contributions

This introducing chapter aims at introducing the context of Federated Learning with constrained communication, and the challenges posed by the use of compression.

We first recall the federated learning framework and describe how compression can be used, the specificities of compression in the context of distributed optimization, as well as the main research directions that have been explored and the contributions made, that are detailed in Chapters 2 and 3.

1.1 Federated Learning and Communication constraints

Federated Learning context. In modern large scale machine learning applications, optimization has to be processed in a distributed fashion, using a potentially large number N of workers. In the data-parallel framework, each worker only accesses a fraction of the data.

Formally, we consider a number of features $d \in \mathbb{N}^*$, and a (convex) cost function $F : \mathbb{R}^d \to \mathbb{R}$. We want to solve the following convex optimization problem:

$$\min_{w \in \mathbb{R}^d} F(w) \text{ with } F(w) = \frac{1}{N} \sum_{i=1}^{N} F_i(w)\,, \tag{1.1}$$

where $(F_i)_{i=1}^N$ is a *local* risk function for the model w on the worker i. Especially, in the classical supervised machine learning framework, we fix a loss ℓ and access, on a worker i, n_i observations $(z_k^i)_{1 \leqslant k \leqslant n_i}$ following a distribution D_i. In this framework, F_i can be either the (weighted) local empirical risk, $w \mapsto (n_i^{-1}) \sum_{k=1}^{n_i} \ell(w, z_k^i)$ or the expected risk $w \mapsto \mathbb{E}_{z \sim D_i}[\ell(w, z)]$. At each iteration of the algorithm, each worker will obtain an *unbiased oracle* on the gradient of the function F_i (typically either by choosing uniformly an observation in its dataset or in a *streaming fashion*, getting a new observation at each step).

In this manuscript, we focus on first-order methods, especially Stochastic Gradient Descent [10, 101]. In the framework we study, a central machine aggregates the computation of the N workers in a synchronized way. This includes both the *distributed* [e.g. 70] and the *federated learning* [introduced in 64, 80] settings. In the federated context,

the functions F_i typically strongly depends on the worker i, a phenomenon referred to as statistical heterogeneity.

Communication constraint and compression. The communication cost has been identified as an important bottleneck in the distributed settings [e.g. 115]. In their overview of the federated learning framework, Kairouz et al. [57] underline (Section 3.5) two ways reduce this cost: 1) Reducing the frequency of communication; and 2) compressing each message exchanged between the workers and the central server. **In the following chapters and references [P1, C7, C6, C8, P5], we focus on the latter.** There is also an abundant literature on the first approach, including [112, 59, and following references], and our works [C1, C10] are part of it.

The communication between the workers to the central server can be compressed in both directions: we refer to *uplink* (worker to server) and *downlink* communication.

Bidirectional compression. While many papers leveraging compression to reduce the communication cost [4, 2, 130, 58, 81, 53, 72, 51] focus on the uplink direction, and this direction has arguably the highest potential to reduce the total runtime there are several reasons to also consider downlink compression. First, the difference between upload and download speeds is not significant enough at all to ignore the impact of the downlink direction (see [95] for an analysis of bandwidth). If we consider for instance a small number N of workers training a very heavy model – the size of Deep Learning models generally exceeds hundreds of MB [16, 55] –, the training speed will be limited by the exchange time of the updates, thus using downlink compression is key to accelerating the process. Moreover, in a framework in which a network of smartphones collaborate to train a large scale model in a federated framework, participants to the training would not be eager to download a hundreds of MB for each update on their phone. Here again, downlink compression appears to be necessary. To encompass all situations, we consider compression in both directions with possibly different compression levels in [C7, P1].

1.1.1 Definitions, examples and assumptions on compression operator.

Numerous techniques have been proposed to perform compression. The simplest way to transmit a d-dimensional vector requires to encode it over $32 \times d$ or $64 \times d$ bits. However, such a precision (a) is not always required in applications, as the quantities exchanged have some intrinsic randomness (b) the communication of such large quantities of data would slow down the exchanges and saturate the bandwidth.

Definition 1.1 (Compression operator)**.** *A compression operator $\mathcal{C}$ on $\mathbb{R}^d$ is a (possibly random) operator $\mathbb{R}^d \to \mathcal{T}$, where $\mathcal{T} \subset \mathbb{R}^d$ is potentially a random set. When the output is a random variable, we say that $\mathcal{C}$ is randomized.*

A compression operator generally satisfies that for most or any x, $\mathcal{C}(x)$ can be transmitted without loss with a number of bits much lower than $64 \times d$. When the compression operator is random, then this statement holds *on average*.

Insights on compression in the context of FL. The study of compression from an information theoretic point of view [6] and the construction of optimal coding schemes for an input distribution has been an active area of research over the second half of the

20th century starting with the work of Shannon [108]. For classical tasks, on images or text, the goal of compression is to be able to recover *almost exactly* the signal from its compressed version. In the context of FL, the objective is very different. We highlight the most important differences:

- **Randomness.** The information exchanged contains inherent randomness, and high-precision or loss-less compression is not necessary. In supervised learning, stochastic gradients are observed and exchanged: they serve as a proxy to the true gradient, but typically suffer from a noise with variance σ^2. A compression precision of the order of σ may thus be sufficient.
- **Unknown distribution.** The distribution of the input source is unknown and has little structure. For example, in the distribution of the gradients (e.g., in a high-dimensional neural network) is far from fully understood.
- **Repeated process.** Finally, the learning algorithm often relies on the repeated communication of multiple elementary elements (e.g., a sequence of stochastic gradients), computed over consecutive iterations and by multiple agents in a distributed setting. It is then crucial to ensure that *on average, i.e., in expectation*, the information transmitted is correct. In the following, we are thus interested in finding compression schemes that are *unbiased*: for any $x \in \mathbb{R}^d$ $\mathbb{E}[\mathcal{C}(x)] = x$.

1.1.2 Unbiased compression operator with relatively bounded variance (UCRBV).

Unbiased compression operators are necessarily randomized. Formally, we will rely on two compression operators $\mathcal{C}_{\rm up}$ and $\mathcal{C}_{\rm dwn}$, that will respectively be applied to any signal transmitted from the local worker to the central server (uplink) and in the other direction (downlink). We have a major focus on operators which have a variance scaling (at most) proportionally to the squared norm of the compressed signal:

A1 (Unbiased compression with relatively bounded variance (UCRBV)). *There exist constants $\omega_{\mathcal{C}}^{\rm up}, \omega_{\mathcal{C}}^{\rm dwn} \in \mathbb{R}_+^*$, such that the compression operators $\mathcal{C}_{\rm up}$ and $\mathcal{C}_{\rm dwn}$ verify the two following properties for all Δ in $\mathbb{R}^d$, for $\mathrm{dir} \in \{\mathrm{up}, \mathrm{dwn}\}$:*

$$\begin{cases} \mathbb{E}[\mathcal{C}_{\rm dir}(\Delta)] = \Delta\,, \\ \mathbb{E}[\|\mathcal{C}_{\rm dir}(\Delta) - \Delta\|^2] \leqslant \omega_{\mathcal{C}}^{\rm dir} \|\Delta\|^2\,. \end{cases}$$

When only one direction is considered, we will simply denote $\mathcal{C}$. Constants $\omega_{\mathcal{C}}^{\rm up}$ and $\omega_{\mathcal{C}}^{\rm dwn}$ parametrize the strength of the compression, and can be considered as *parameters* of the algorithm, as the compression levels can be chosen depending on the setting. This assumption will be made and studied in chapters Chapters 2 and 3.

Variance reduction via unbiased compressors. In multiple applications under consideration, multiple agents, measurement units or sources may communicate compressed values of a similar quantity. The importance of unbiasedness is then most visible: indeed, the error of the averaged message decreases with the number of measurements N. The simplest situation is the case in which each communication corresponds to an *independent and identically distributed compression operators* $(\mathcal{C}_k)_{k=1}^N$ of the *same vector* $x \in \mathbb{R}^d$. The aggregation $N^{-1}\sum_{k=1}^N \mathcal{C}_k(x)$, then admits a bias-variance decomposition of its quadratic error: $\mathbb{E}[\|N^{-1}\sum_{k=1}^N \mathcal{C}_k(x) - x\|^2] = \|\mathbb{E}[\mathcal{C}_1(x)] - x\|^2 + N^{-1}\mathbb{E}[\|\mathcal{C}_1(x) - \mathbb{E}[\mathcal{C}_1(x)]\|^2]$. In words, the variance of the aggregated vector is reduced by a factor N^{-1} when averaging the

N messages, while the bias is independent of N. For example, if we use independent unbiased compressors satisfying A 1 with a constant ω, $N^{-1}\sum_{k=1}^{N}\mathcal{C}_k(x)$ satisfies A1 with a constant ω/N. A similar dependence is observed in subsequent convergence results, e.g., in Chapters 2 and 3, in which the impact of the uplink compression will be reduced by a factor N, while biased compression operators do not benefit from such a reduction [38].

1.2 Construction of UCRBV and examples

In this section, we review the main approaches to perform unbiased compression. We consider a possibly random mapping $\mathcal{C}:\mathbb{R}^d\to\mathcal{T}$, following Definition 1.1. *Compressing* the information means that any element of the set $\mathcal{T}$ can be stored or communicated using a limited number of bits. In most situations, $\mathcal{T}$ is (or has a one-to-one mapping to) either a finite set Ψ_M (called a *codebook*) with cardinality M, a low dimensional space $\mathbb{R}^h$, $h\ll d$, or the combination of both $\mathbb{R}^h\times\Psi_M$. The latest requires $32h+\log_2(M)$-bits to be transmitted in a standard precision regime.

As a consequence, compression operators typically belong to two main categories: quantization-type, that rely on a codebook Ψ_M [as in 4, 107, 136, 128, 100, 53] and sparsification-type, that correspond to projections onto (random) spaces (of average dimension $h\ll d$) [as in 114, 3, 5, 60].

In the following, we describe the most important examples through a principled approach.

1.2.1 Sparsification: compression via (random) projections

The simplest way of transforming the vector to reduce the number of bits to communicate is to apply a (random) projection onto a smaller dimensional subspace of dimension $h\ll d$. Noticeable examples include projections that do not depend on x:

Definition 1.2 (Rand-h)**.** *We transmit a subset of h coordinates out of d. This corresponds to projecting orthogonally on a subspace selected uniformly among C_h^d possible subspaces, generated by exactly h canonical vectors. In order to obtain un-biasedness, the random projection is then scaled by a factor d/h.*

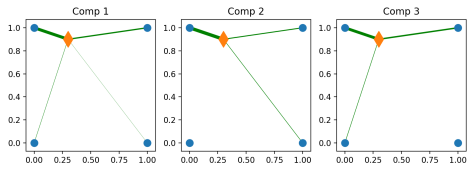
Definition 1.3 (p-sparsification)**.** *We transmit each coordinate with probability p, independently of each other. This corresponds to projecting onto a random subspace with dimension $\mathcal{B}(p,d)$ a binomial law of parameters (p,d). In order to obtain un-biasedness, the random projection is then scaled by a factor p^{-1}.*

Definition 1.4 (Gaussian projection)**.** *We sample a Gaussian matrix G of size $d\times h$ and transmit $G^\top x\in\mathbb{R}^h$, then reconstruct $G(G^\top G)^{-1}G^\top x\in\mathbb{R}^d$.*

Definition 1.5 (Partial-participation)**.** *We transmit the full input x/p with probability p.*

The latest example is of particular interest as device partial participation is a well identified challenge in distributed systems. If all devices participate independently of each other with probability p, then this can be seen as identified as a particular compression scheme.

Those compression schemes are not typically the most efficient in practice as they result in transmitting with high precision a small part of x and completely ignoring the rest.



Ψ_M	(0,0)	(1,0)	(0,1)	(1,1)	QE
$\mathcal{C}_1 : \lambda_i(x_0)$	0.07	0.63	0.03	0.27	0.3
$\mathcal{C}_2 : \lambda_i(x_0)$	0	0.1	0.7	0.2	0.3
$\mathcal{C}_3 : \lambda_i(x_0)$	0.1	0	0.6	0.3	0.3

Figure 1.1: Three possible compression schemes for a vector $x_0 = (0.3, 0.9)$ on a given codebook Ψ_M. The orange diamond corresponds to the point to compress, and the blue points to the codebook. The width of the green line corresponds to the weight assigned to each codeword. . The table provides the exact weights and quadratic error (QE)

However, their linearity $\mathcal{C}(x_1) - \mathcal{C}(x_2) = \mathcal{C}(x_1 - x_2)$ can be helpful to enhance the stability of algorithms see e.g., Chapter 2.

1.2.2 Unbiased quantization on a deterministic codebook.

The other main approach to obtain an unbiased compression scheme consists in rescaling the input vector, then selecting an element of a codebook. For a given codebook, we first introduce, this technique to quantize a vector x in the convex hull of the codebook elements $\mathrm{Conv}(\Psi_\mathrm{M})$.

Unbiased quantization on a deterministic codebook on its convex hull. We first introduce the set of unbiased compression on a codebook.

Definition 1.6 (Unbiased quantization on a codebook)**.** *We consider a codebook Ψ_M of cardinal $M \subset \mathbb{R}^M$ and $x \in \mathcal{X} = conv(\Psi_M)$. An unbiased compression scheme with codebook Ψ_M is a random mapping:*

$$\begin{aligned} \mathcal{C}_{\Psi_M} : \mathcal{X} &\to \Psi_M \\ x &\mapsto \psi_i \quad \textit{with probability } \lambda_i(x)\,. \end{aligned}$$

such that for all $x \in \mathcal{X}$,

$$\mathbb{E}\left[\mathcal{C}_{\Psi_M}(x)\right] = \sum_{i=1}^{M} \lambda_i(x)\psi_i = x \tag{1.2}$$

We denote $\mathrm{UVQ}(\Psi_\mathrm{M})$ the set of unbiased vector quantization operators with codebook Ψ_M. The quadratic error of such a compression operator at a point $x \in \mathcal{X}$ is then

$$\mathbb{E}[\|x - \mathcal{C}_{\Psi_M}(x)\|^2] = \sum_{i=1}^{M} \lambda_i(x)\|\psi_i - x\|^2\,. \tag{1.3}$$

In this class, the codebook Ψ_M is considered deterministic, and the only randomness comes from the sampling of the codeword ψ_i. The resulting codeword ψ_i can be encoded via $\log M$ bits. Such a code is optimal under the assumption that the dictionary atoms are uniformly sampled (which corresponds to an assumption on the input distribution $\mathcal{X}$).

Remark 1.7. *For a given codebook, several such compression schemes may exist in* $\mathrm{UVQ}(\Psi_\mathrm{M})$*. In Figure 1.1 we propose three possible unbiased quantization on $\Psi_4 = \{(0,0),(0,1),(1,0),(1,1)\}$ for an example vector $x_0 = (0.3, 0.9) \in [0;1]^2 = \mathrm{Conv}(\Psi_4)$.*

Schemes in the class UVQ-Ψ_M are restricted to the set $\mathcal{X} = \mathrm{Conv}(\Psi_\mathrm{M})$. E.g., for the codebook in Figure 1.1, we have $\mathcal{X} = B_\infty(0,1)$ in dimension 2. In order to compress any vector $x \in \mathbb{R}^d$, we :

- Rescale x such that $x/\|x\|_p \in \mathcal{X}$.
- Transmit $\|x\|_p$ and $\mathcal{C}(x/\|x\|_p, \Psi_M)$.
- Output $\|x\|_p \times \mathcal{C}(x/\|x\|_p, \Psi_M)$.

One observes that the unbiased property is preserved, as $\mathbb{E}[\|x\|_p \times \mathcal{C}(x/\|x\|_p, \Psi_M)] = x$. This can be seen as a gain-shape transform.

Gain-shape and bin splitting Before quantization, a typical strategy [35, Chap 3 Section12.10] is to separate a vector $x \triangleq g.s \in \mathbb{R}^D$ into $g \triangleq \|x\|_p$, also known as gain, and $s \triangleq \frac{x}{\|x\|_p}$ which is also referred to as the shape. Moreover, the shape $s \in \mathbb{R}^D$ is itself partitioned in *bins* $s = (s_i)_{i \leqslant k} \in (\mathbb{R}^d)^k$, so that $D = d \times k$. Each bin (aka *bucket*) is quantized separately in $\mathbb{R}^d$.

We have the straightforward link between the variance of an operator in $\mathrm{UVQ}(\Psi_\mathrm{M})$ and A 1.

Lemma 1.8 (Gain-shape transform and A1.)**.** *If $\mathcal{B}_2(1) \subset \mathrm{Conv}(\Psi_\mathrm{M})$ and if the quadratic error of $\mathcal{C}_{\Psi_M}$ is uniformly upper bounded on $\mathrm{Conv}(\Psi_\mathrm{M})$ by a constant ω, then the operator performing the gain shape transform $Q_{\Psi_M} : x \mapsto \|x\|_2 \mathcal{C}_{\Psi_M}\left(\frac{x}{\|x\|_2}\right)$ satisfies A1 with constant ω.*

Remark 1.9 (Communication cost for unconstrained UVQ)**.** *Combining gain-shape transformation with an unbiased compression onto a codebook Ψ_M corresponds to a compression operator which output is in $\mathbb{R} \times \Psi_M$. The communication cost is then, a priori, 32-bits to transmit in a loss-less fashion $\|x\|_p$ and $\log(M)$ bits to transmit the codeword index $i \in [M]$. In many applications, it is reasonable to also quantize the gain $\|x\|_p$ with a one dimensional scalar-quantizer.*

This lemma has been widely used in the design of compression schemes for machine learning applications. We now present two techniques to obtain such UVQ. The most widely used consists on compressing independently each coordinate, which is requires to use a codebook on a grid.

Grid codebooks and scalar quantization (SQ) over $\mathcal{B}_\infty(0,1)$ A *scalar quantization* scheme is such that each coordinate is compressed independently, or more formally that for any $x \in \mathcal{X}$, $(\mathcal{C}_{Q,\Psi_M})_{i=1,\dots,d}$ are mutually independent.

Definition 1.10 (Grid codebook)**.** *We say that a codebook $\Psi_M \subset \mathbb{R}^d$ is*

- *a* grid *if $\Psi_M = \mathcal{L}^d$, where $\mathcal{L} = \{\ell_0, \dots, \ell_L\}$ is a finite subset of $\mathbb{R}$ with cardinal $L+1$.*
- *a* evenly-spaced grid *if furthermore $\mathcal{L} = \{\ell_0 + k\frac{\ell_L - \ell_0}{L}, 0 \leqslant k \leqslant L\}$.*

Working with a grid codebook allows to quantize independently each coordinate to obtain an unbiased vector quantization scheme,

Definition 1.11 (Scalar quantization for a grid $\mathcal{L}^d$)**.** *A* Scalar quantization *method with codebook $\Psi_M = \mathcal{L}^d$ is the quantization operator in $\mathrm{UVQ}(\Psi_\mathrm{M})$ such that, for any $x \in [\ell_0; \ell_L]^d$ and $i \in [d]$:*

$$(\mathcal{C}_{Q,\Psi_M}(x))_i = \ell_{j_x^i} + X_i(\ell_{j_x^i+1} - \ell_{j_x^i}), \tag{1.4}$$

where j_x^i is the only index such that $\ell_{j_x^i} \leqslant x_i < \ell_{j_x^i+1}$, and $X_i \sim \mathcal{B}(\frac{x_i - \ell_{j_x^i}}{\ell_{j_x^i+1} - \ell_{j_x^i}})$, and $(X_i)_{1 \leqslant i \leqslant d}$ are mutually independent. If $x_i = \ell_L$, then we use by convention $j_x^i = L-1$.

Scalar quantizer were originally used in signal processing [35, Chap 2] and re-introduced for applications in Federated Learning in the celebrated QSGD paper by Alistarh et al. [4].

Example 1.12. *In most implementations, the codebook is an evenly spaced grid with* resolution *s:*

$$\Psi_M = \mathcal{L}^d_{\text{reg,s}} = \left\{ \frac{k}{2^s},\ k \in \{-2^s, \dots, 2^s\} \right\}^d.$$

Ψ_M has a log-cardinality of $\log_2(M) = d\log_2(2^{s+1}+1) \simeq d(s+1)$

Such a codebook ensures that $\mathcal{B}_\infty(0,1) \subset \text{Conv}(\Psi_\text{M})$, and the compression scheme can thus be applied on any ball $\mathcal{B}_p(0,1)$. The strongest compression corresponds to $s=0$, for which $\Psi_M = \{-1,0,1\}^d$, and each coordinate is compressed onto $\log_2(3)$-bits. In Figure 1.1, SQ corresponds to the first line $\mathcal{C}_1$.

Computation: SQ has several noticeable advantages. The main advantage is that the output can be computed efficiently and in a trivially parallelizable fashion over the d coordinates. Moreover, the grid can be optimized and the cost of storing it is negligible. The main drawback is that the number of atoms in the codebook scales exponentially with the dimension: in other words, the compression rate cannot be stronger that 1-bit per coordinate (thus a maximal compression rate of 32).

Remark 1.13 (Quadratic Error decomposition)**.** *The quadratic error of $\mathcal{C}_{Q,\mathcal{L}^d}$ decomposes over coordinates:*

$$\mathbb{E}\left[\|x - \mathcal{C}_{Q,\mathcal{L}^d}(x)\|^2\right] = \sum_{i=1}^d \mathbb{E}\left[(x_i - \mathcal{C}_{Q,\mathcal{L}}(x_i))^2\right]$$

Remark 1.14 (Optimal scalar grid $\mathcal{L}$.)**.** *The best scalar grid has been studied for $\mathcal{X} \in \{\mathcal{B}_2, \mathcal{B}_\infty\}$.*

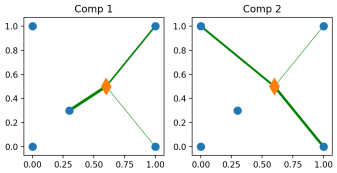
- *For $\mathcal{X} = \mathcal{B}_\infty(1)$, the evenly spaced grid $\mathcal{L}_{\text{reg,s}}$ corresponds to grid resulting in the minimal worst-case quadratic error. It is clear from Remark 1.13 and because in dimension 1,*

$$\sup_{x_i \in [-1;1]} \mathbb{E}\left[(x_i - \mathcal{C}_{Q,\mathcal{L}}(x_i))^2\right] = \frac{\sup_{j\in[L]}(\ell_{j+1}-\ell_j)^2}{4}.$$

- *This grid is also optimal in terms of averaged performance against a uniform distribution on $\mathcal{B}_\infty(1)$, i.e., $x \sim \mathcal{U}[-1;1]^d$.*
- *For $\mathcal{X} = \mathcal{B}_2(1)$, such a grid is not worst case optimal anymore, on the contrary, a geometrically spaced grid $\mathcal{L}_\tau = \{\pm\tau^{-j}, j \in [L]\}$ is showed to be optimal in [99].*

Remark 1.15 (Codewords sampling distribution and optimal encoding.)**.** *In most situations, it is impossible to compute the probability that each codeword is sampled without making distributional assumptions on the input distribution, and the general practice is to encode each codeword with a code of fixed length* $\log_2(M)$*. This corresponds to the optimal code if $x \sim \mathcal{U}(\mathcal{B}_\infty(0,1))$, and $\Psi_M = \mathcal{L}^d_{\text{reg,s}}$.*

One noticeable exception is the case of $\mathcal{X} = \mathcal{B}_2(1)$. Indeed, for any $x \in \mathcal{B}_2(1)$, $\mathcal{C}_{Q,\mathcal{L}^d_{\text{reg,s}}}$ is unevenly distributed over $\mathcal{L}^d_{\text{reg,s}}$. For example for $s=0$, $\mathbb{E}[\|\mathcal{C}_{Q,\Psi_M}(x)\|_0] = \|x\|_1 \leqslant \sqrt{d}$, thus, the expected number of non-zero coordinates of the codeword is bounded by $\sqrt{d}$ for any x. As a consequence, for any distribution on $\mathcal{B}_2(1)$, it is possible to encode $\mathcal{C}_{Q,\Psi_M}(x)$ in $2\sqrt{d}$ bits.



Ψ_M	(0,0)	(0.3,0.3)	(1,0)	(0,1)	(1,1)	QE
$\mathcal{C}_1 : \lambda_i(x_0)$	0	0.57	0.1	0	0.33	0.25
$\mathcal{C}_2 : \lambda_i(x_0)$	0	0	0.5	0.4	0.1	0.49

Figure 1.2: Two possible compression schemes for a vector $x_0 = (0.6, 0.95)$ on a given codebook Ψ_M with 5 atoms. The orange diamond corresponds to the point to compress, and the blue points to the codebook. The width of the green line corresponds to the weight assigned to each codeword. The table provides the exact weights and quadratic error (QE).

While scalar quantization has noticeable advantages, several questions remain. Especially: (1) does there exist a better convex decomposition to minimize the quadratic error? (2) can we find a better codebook with same cardinality that reduces the quadratic error? and (3), more crucially, how can one achieve stronger compression (going below the 1-bit per coordinate threshold, thus excluding grid codebooks)? *Delaunay quantization* provides a systematic way to minimize the quadratic error among $\mathrm{UVQ}(\Psi_\mathrm{M})$ for a general codebook.

Delaunay Quantization (DQ) The Delaunay quantization operator is defined, for a codebook Ψ_M and $x \in \mathrm{Conv}(\Psi_\mathrm{M})$ as the unbiased vector quantizer over Ψ_M minimizing *at any point* the quadratic error.

$$\mathcal{C}_{D,\Psi_M}(x) = \left(\underset{\mathcal{C} \in \mathrm{UVQ}(\Psi_\mathrm{M})}{\arg\min} \ \mathbb{E}[\|x - \mathcal{C}_{\Psi_M}(x)\|^2] \right)(x) \tag{1.5}$$

For any x, this is equivalent to solving the following problem over the M dimensional-probability simplex - finding the best $\lambda_{D,\Psi_M}(x)$ such that:

$$\lambda_{D,\Psi_M}(x) \in \underset{\substack{\lambda \in [0,1]^M, \sum_{i=1}^M \lambda_i = 1 \\ \sum_{i=1}^M \lambda_i \psi_i = x}}{\arg\min} \sum_{i=1}^M \lambda_i \|\psi_i - x\|^2 . \tag{1.6}$$

DQ was introduced in [129], as dual-quantization. The main advantage of DQ is that the quadratic error is minimized at any point x. Its main bottleneck is that computing the optimal $\lambda_{D,\Psi_M}(x)$ is not straightforward. However, eq. (1.6) is a convex problem and can be solved efficiently. In most situations, Delaunay quantization corresponds to partitioning the convex hull of Ψ_M into polytopes supported by (at most) $d+1$ codewords, $\mathrm{Conv}(\{\psi_{\mathrm{i_j}}, \mathrm{j} \in [\mathrm{d}+1]\})$. An example of DQ is given in Figure 1.2.

Delaunay and Scalar quantization. Interestingly, *for a grid codebook $\mathcal{L}^d$, any* decomposition over the 2^d points $\prod_{i=1}^d \{\ell_{j_x^i}, \ell_{j_x^i+1}\}$ results in the exact same quadratic error. For example in Figure 1.1, one can observe that the quadratic error is the same for all three quantizations.

Lemma 1.16 (Scalar quantization is Delaunay optimal on $\mathcal{L}^d$)**.** *For a grid codebook $\mathcal{L}^d$, the scalar quantization on the grid (as defined in Definition 1.11) is a Delaunay decomposition, i.e., satisfies Equation* (1.5).

As a consequence, when using a scalar grid (evenly-spread) or not, SQ achieves optimality in the sense the the convex decomposition obtained by the decomposing coordinates independently

The techniques described above in a principled way constitute the most popular approaches to perform compression. UCRBV can also be combined, as for any two independent operators $\mathcal{C}_1, \mathcal{C}_2$ satisfying A1 with constant ω_1, ω_2, $\mathcal{C}_1 \circ \mathcal{C}_2$ satisfies A1 with constant $(1+\omega_1)(1+\omega_2) - 1$.

Contribution. In [P5], we introduce new compression schemes based on Delaunay compression with non grid codebooks. We also introduce a novel compression approach, based on codebook randomization, that enables to reduce by a factor 2 or 3 the ω constant in A1 for a given codebook cardinality.

1.3 Assumptions on the optimization problem

In the following paragraphs, we summarize some of the assumptions requires to obtain convergence guarantees in Part I. In Section 1.3.1, we present the assumptions typically made on the functions to be optimized or on the sequence of gradient oracles, and in Section 1.3.2 the assumptions on the task heterogeneity.

We consider the optimization problem Equation (1.1), distributed over N agents. In most of part I, unless explicitly mentioned, we will assume all functions to be differentiable and the optimization domain to be $\mathbb{R}^d$.

1.3.1 Assumptions on the objective function and stochastic gradients

Assumptions on the optimization problem Equation (1.1). In the following chapters on optimization, we will make some of the following classical assumptions on $F : \mathbb{R}^d \to \mathbb{R}$. Assumptions are regrouped early in this chapter as they will be used at several places, but not all assumptions will be made simultaneously. For example, we will provide results both in the non convex case and in the convex case.

A2 (Convexity). *F is convex, that is for all vectors w, v in $\mathbb{R}^d$: $F(v) \geqslant F(w) + (v-w)^T \nabla F(w)$.*

Moreover, to obtain linear convergence rates, we will rely on strong convexity.

A3 (Strong convexity). *F is μ-strongly convex, that is for all vectors w, v in $\mathbb{R}^d$: $F(v) \geqslant F(w) + (v-w)^T \nabla F(w) + \frac{\mu}{2}\|v-w\|_2^2$.*

Remark that we do not typically need each F_i to be convex or strongly-convex, but only F. Moreover, for many of the results (e.g., Theorem 2.1) the assumption can be made only for $v = w_*$. Finally, we will rely on the smoothness of F:

A4 (Smoothness). *F is twice continuously differentiable, and is L-smooth, that is for all vectors w, v in $\mathbb{R}^d$: $\|\nabla F(w) - \nabla F(v)\| \leqslant L\|w - v\|$.*

Assumptions on the stochastic oracle model. We will rely on stochastic optimization methods. That means that over multiple iterations $k \in \mathbb{N}$, each worker $i \in [\![1; N]\!]$ will query a stochastic oracle of the gradient, denoted g_k^i, such that, if we denote $(F_k)_{k\geqslant 0}$ a filtration such that for k, and all $i \in [\![1; N]\!]$, g_k^i is $\mathcal{F}_k$ measurable

A5 (Unbiased gradients). *For all $k \in \mathbb{N}$, and all $i \in [\![1; N]\!]$:*

$$\mathbb{E}[g_k^i | \mathcal{F}_{k-1}] = \nabla F_i.$$

We will also assume in Chapter 3 that the variance of the noise is uniformly controlled. As the oracle is generally computed on a mini-batch of size b, that results in a reduced variance, we incorporate this fact in the assumption:

A6 (Bounded noise variance on stochastic gradients oracle). *The noise over stochastic gradients for a mini-batch of size b, is uniformly bounded: there exists a constant $\sigma \in \mathbb{R}_+$, such that for all k in $\mathbb{N}$, for all i in $[\![1, N]\!]$ and for all w in $\mathbb{R}^d$ we have: $E[\|g_k^i(w) - \nabla F(w)\|^2 | \mathcal{F}_{k-1}] \leqslant \sigma^2/b$.*

However, in convex optimization, this assumption is typically a bit too strong, and we prefer to only assume the variance to be controlled at the optimal point w_*. The following assumption is made in Chapter 2.

A7 (Noise over stochastic gradients computation). *The noise over stochastic gradients at the global optimal point, for a mini-batch of size b, is bounded: there exists a constant $\sigma_* \in \mathbb{R}$, s. t. for all k in $\mathbb{N}$, for all i in $[\![1, N]\!]$, we have a.s.:* $\mathbb{E}[\|g_k^i(w_*) - \nabla F_i(w_*)\|^2 | \mathcal{F}_{k-1}] \leqslant \frac{\sigma_*^2}{b}$.

The constant σ_*^2 is null, e.g. if we use deterministic (batch) gradients, but also more general settings, e.g. in interpolation regimes, for example for a well specified linear regression problem in which all observations satisfy $Y_j = X_j^\top w_*$.

In order to obtain fast convergence rates for stochastic optimization with bounded noise variance at *only* the optimal point, we will need to control the average regularity of the stochastic gradients. Below, we introduce cocoercivity [see 137, for more details about this hypothesis].

A8 (Cocoercivity of stochastic gradients (in quadratic mean)). *We suppose that for all k in $\mathbb{N}$, stochastic gradients functions $(g_k^i)_{i \in [\![1,N]\!]}$ are L-cocoercive in quadratic mean. That is, for k in $\mathbb{N}$, i in $[\![1, N]\!]$ and for all vectors w, v in $\mathbb{R}^d$, we have:*

$$\mathbb{E}[\|g_k^i(w) - g_k^i(v)\|^2] \leqslant L \langle \nabla F_i(w) - \nabla F_i(v) \mid w - v \rangle \ .$$

E.g., this is true under the much stronger assumption that stochastic gradients functions $(g_k^i)_{i \in [\![1,N]\!]}$ are *almost surely* L-cocoercive, i.e.: $\|g_k^i(w) - g_k^i(v)\|^2 \leqslant L \langle g_k^i(w) - g_k^i(v) \mid w - v \rangle$. Remark that this assumption implies that all $(F_i)_{i \in [\![1,N]\!]}$ are L-smooth. Finally, we point that other general assumptions on the oracle's regularity could be made, e.g. expected smoothness [46].

1.3.2 Assumptions on task heterogeneity

The data distribution depends on each worker: there are multiple ways to quantify this heterogeneity, between the functions. For example, it is possible to compare the function values F_i, or, if they exist the values of the individual optimal points $w_i^* := \arg\min_{w \in \mathbb{R}^d} F_i(w)$. In this manuscript, we will rely in Chapters 2 and 3 on an assumption that describes the difference between the gradients.

More precisely, in the convex setting in which F admit a global minimum w^*, the local gradient at the optimal point $\nabla F_i(w_*)$ may not vanish. We consider the following quantity:

A9 (Bounded gradient at w_*)**.** *There exists a constant $B \in \mathbb{R}_+$, s.t.:*

$$\frac{1}{N}\sum_{i=0}^{N} \|\nabla F_i(w_*)\|^2 \leqslant B^2 \,.$$

This is not, strictly speaking, an assumption, but more a definition of a constant B^2, that quantifies the amount of heterogeneity. Especially, in the homogeneous framework (e.g., in the streaming *i.i.d. setting* – $D_1 = \cdots = D_N$ and $F_1 = \cdots = F_N$) the assumption is satisfied with $B = 0$.

Remark that there can exist heterogeneous frameworks in which all functions are not equal, yet $w_i^* = w_j^* = w*$ for any $i, j \leqslant N$. In such case, we can have $B^2 = 0$. This is not an issue, as we will show that such a condition is sufficient to almost recover the convergence of the homogeneous case.

This assumption requires the existence of a global optimal point, so is best suited for the convex case. It can be extended to the following assumption [59]:

A10 ((G, B)-BGD or bounded gradient dissimilarity)**.** *There exist constants $B \geqslant 0$ and $G \geqslant 1$ such that $\forall w \in \mathbb{R}^d$*

$$\frac{1}{N}\sum_{i=1}^{N} \|\nabla F_i(w)\|^2 \leqslant B^2 + G^2\|\nabla F(w)\|^2.$$

If $\{F_i\}$ are convex, we can relax the assumption to $\forall w \in \mathbb{R}^d$

$$\frac{1}{N}\sum_{i=1}^{N} \|\nabla F_i(w)\|^2 \leqslant B^2 + 2\beta G^2 \left(F(w) - F^\star\right).$$

1.4 Summary of the main results - main challenges and insights for algorithms with compression

For a first order algorithm, adding compression raises multiples interconnected questions. I choose to highlight in the manuscript contributions addressing those various questions.

1. How does a naive incorporation of bi-compression into a learning algorithm **impact convergence?**

↦ In Chapter 2, we show that incorporating compression into algorithms results in an increased variance, especially in the presence of worker heterogeneity, and quantify the resulting convergence in the case of *bi-directional* compression. Relying on a convergence in Wasserstein distribution, we describe the exact asymptotic variance of the resulting process. (Based on [P1])

2. Can we adapt the learning algorithm to **reduce the impact of heterogeneity?**

↦ In Chapter 2, we also show how to adapt the learning algorithm to recover with compression a convergence (nearly) similar to the one in the homogeneous setting. We rely on a client-wise control variate, that enables to learn the client dissimilarity and annihilate its impact on convergence. We provide convergence rates for the convex case and for a *federated EM algorithm*. (Based on [C6, P1, C8, C10])

3. Can we modify the algorithm to (nearly) **recover the rate of un-compressed algorithms?**

↦ In Chapter 3, we consider a bi-directional compression framework. We introduce the concept of preserved model: when communicating to the workers, the central server sends a compressed version of its update, but makes a different update locally, without suffering from the downlink compression. We show that this annihilates the impact of downlink compression, up to negligible second order terms. (Based on [C7])

4. Open directions: can we go **beyond worst-case quadratic bound** on the compression operator's variance?

↦ In the Open directions chapter , I briefly describe some ongoing lines of work, that enable to obtain a refined understanding of the behavior of compression operators. Especially, we study:

- **Second order moments of the compression operator:** several compression operators that satisfy the same assumption A1 can have different asymptotic convergence rates.
- **Higher order moments of the compression operator.** We show how two compression operators that satisfy the same assumption A1 can have different higher order moments and how this impacts convergence.
- **Lack of regularity.** A typical feature of quantization is that it degrades the regularity of the process: typically $\mathbb{E}[\|\mathcal{C}(x) - \mathcal{C}(y)\|^2] \gg \|x - y\|^2$ when $x - y \to 0$. We show how this impacts the *stability of the algorithm*, to obtain generalization bounds.

Each of these questions and the (partial) answers we provide are part of a broad community effort to better understand the impact of compression. A short summary of these chosen contributions is given in the next chapters.

2

Federated learning with heterogeneity and compression

This chapter is based on [P1], *Artemis: tight convergence guarantees for bidirectional compression in Federated Learning*, C. Philippenko and **A.D.**; and [C6] *Federated-EM with heterogeneity mitigation and variance reduction*, **A.D.**, G. Fort, E. Moulines, G. Robin, Neurips 2021;

Both these papers tackle Federated optimization with compression and heterogeneity, and investigate the interaction between those two aspects.

References [C10, C8] are also relevant, and are briefly mentioned in the chapter.

2.1 Intuition: link between heterogeneity and compression

The relation between heterogeneity and compression can be understood intuitively. Let us consider the Federated optimization problem (1.1) with heterogeneity and in a convex case. A schematic illustration is given in Figure 2.1. Each function F_i may have a different minimizer w_*^i, which makes w_* a non-equilibrium point for each function F_i, in other words, $\nabla F_i(w_*) \neq 0$ (while $\nabla F(w_*) = 0$).

We consider compression operators $\mathcal{C}$ satisfying A1: this means that their variance scales proportionally to the squared norm of the compressed vector. A simple example to grasp the impact of compression with heterogeneity is the one of compressed distributed GD, i.e.

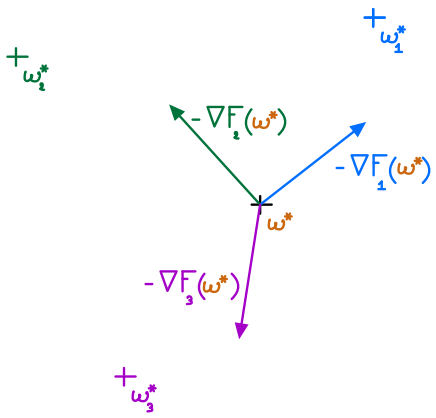


Figure 2.1: Schematic illustration of FL in a convex heterogeneous setting

$$w_t = w_{t-1} - \gamma \sum_{i=1}^{N} \mathcal{C}(\nabla F_i(w_{t-1})).$$

While without compression, this algorithm is GD and would converge at a linear rate for strongly convex functions, introducing compression induces a noise, whose variance scales with $\|\nabla F_i(w_*)\|^2$, asymptotically, if $w_t \to w_*$. This thus strongly hinders convergence.

A particularly simple case is the one of *partial participation*, that can be seen as a compression scheme (see Section 1.2). If only one agent i_t out of the $N = 3$ agents participate at each iteration t, the algorithm $w_t = w_{t-1} - \gamma \nabla F_{i_t}(w_{t-1})$ will not be converging for any constant learning rate $\gamma > 0$, contrary to the full participation regime (i.e., uncompressed algorithm).

In order to recover (linear) convergence, a sufficient solution would be to correct each function F_i such that all functions have the *same* optimal point. This can be achieved by adding a linear correction to each function F_i, setting $\tilde{F}_i : w \mapsto F_i(w) - \langle \nabla F_i(w_*), w \rangle$. Such a transformation does not change the average objective as $\sum_{i=1}^N F_i = \sum_{i=1}^N \tilde{F}_i$, but the resulting functions all have w_* as a critical point, as for all $i \leqslant N$, $\nabla \tilde{F}_i(w_*) = 0$.

Obviously, the values $\nabla F_i(w_*)$ are unknown (as finding w_* is the goal of our approach). In practice, at iteration $k \in \mathbb{N}$ we introduce a *control variate* h_t^i (sometimes coined *memory*), that asymptotically approaches $\nabla F_i(w_*)$, and work on the function $\tilde{F}_{i,t} : w \mapsto F_i(w) - \langle h_t^i, w \rangle$. This corresponds to subtracting h_t^i to the (stochastic) gradients of F_i before compression.

More generally, this can be understood as a form of client-wise variance reduction. Variance reduction ideas originate in the work on Schmidt et al. [106].

In the following, we first present some of the results obtained in [P1], in the case of convex optimization, then some of the results of [C6], that rely on a similar approach to obtain rates for a Federated EM algorithm with compressed communication.

Summarized contributions: The two contributions presented in Sections 2.2 and 2.3 rely on this central idea. In Section 2.2, the first contribution is to quantify how a naive implementation bi-directional compression impacts the convergence of stochastic gradient descent, in a convex optimization regime. This highlights that compression is particularly detrimental to the convergence in the heterogeneous case. We then show how incorporating control variates into the algorithm, corresponding to a client-wise variance reduction scheme, enables to recover a convergence rate similar to the one in the homogeneous case. In Section 2.3, we describe how to incorporate compression into a federated EM algorithm, by acting on the space of sufficient statistics. Although the optimization problem is non-convex, we show that similar control variates enable to obtain convergence rates that do not suffer from the task heterogeneity.

Duality with local iterations, [C10]. A very similar phenomenon happens when using *local iterations* to limit the communication cost. The behavior or Local-SGD (also coined FedAvg) in the homogeneous case [112], [C1], is substantially better than in the heterogeneous case [71]. Intuitively, if one performs too many local iterations, then each workers' sequence of iterates converges to the local optimum w_*^i, and the aggregation $\bar{w}_* := \frac{1}{N} \sum_{i=1}^N w_*^i$ differs from w_*. This phenomenon is called *client-drift*. In that context, Karimireddy et al. [59] proposed to introduce control variates to correct the heterogeneity. Again, the control variates correspond to linear adjustment of each objective

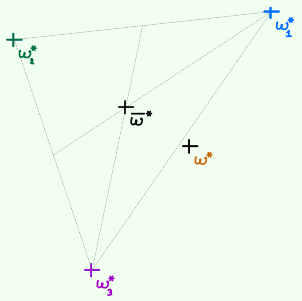


Figure 2.2: Schematic illustration of client drift

function that enable to recover the same critical point asymptotically. Scaffold algorithm has become very popular in practice.

In [C10], *Differentially private federated learning on heterogeneous data*, M. Noble, A. Bellet, and **A.D.**, AISTATS 2022, we extend Scaffold to incorporate differential privacy, providing precise privacy/utility tradeoffs for algorithms with local iterations, heterogeneity, and privacy.

2.2 Artemis: unified framework for SGD with bi-compression

In this section, we consider the problem of learning in a distributed or federated setting with communication constraints problem (1.1) with convex losses. We assume that there exist at least one optimal point which we denote w_*.

A stochastic gradient g_{k+1}^i is provided at iteration k in $\mathbb{N}$ to the device i in $[\![1, N]\!]$. This function is then evaluated at point w_k: we will use $g_{k+1}^i = g_{k+1}^i(w_k)$ and $g_{k+1,*}^i = g_{k+1}^i(w_*)$ to denote the stochastic gradient vectors at points w_k and w_* on device i. In the classical centralized framework (without compression), for a learning rate γ, SGD corresponds to:

$$w_{k+1} = w_k - \gamma \frac{1}{N} \sum_{i=1}^{N} g_{k+1}^i. \tag{2.1}$$

In Artemis, to alleviate the communication cost of distributed algorithms, we focus on bi-directional compression, as introduced in Section 1.1, and use two compression operations we denote $\mathcal{C}_{\rm up}$ and $\mathcal{C}_{\rm dwn}$, satisfying A1. Moreover, we focus on a *heterogeneous* setting, as described in Section 1.3.2, and denote $h_*^i = \nabla F_i(w_*)$, for i in $[\![1, N]\!]$.

To reduce the impact of heterogeneity, we show that it is necessary to incorporate a *memory* process that reduces the size of the signal to compress, and consequently the error [81, 72]. Instead of directly compressing the gradient, we first approximate it by the memory term and, afterwards, we compress the difference. As a consequence, the compressed term tends in expectation to zero, and the error of compression is reduced. At each iteration, we thus have the following steps:

1. First, each active local node sends to the central server a compression of gradient differences: $\widehat{\Delta}_k^i = \mathcal{C}_{\rm up}(g_{k+1}^i - h_k^i)$, and updates the *memory term* $h_{k+1}^i = h_k^i + \alpha \widehat{\Delta}_k^i$ with $\alpha \in \mathbb{R}^*$. The server recovers the approximated gradients' values by adding the received term to the memories kept on its side.

2. Then, the central server sends back the compression of the sum of compressed gradients: $\Omega_{k+1} = \mathcal{C}_{\rm dwn}\left(\frac{1}{N}\sum_{i=1}^N \widehat{\Delta}_k^i + h_k^i\right)$.

Constants $\gamma, \alpha \in \mathbb{R}^* \times \mathbb{R}_+$ are learning rates for respectively the iterate sequence and the memory sequence. The update is thus given by, as summarized in Figure 2.3.:

$$\left\{ \begin{array}{l} \forall i \in [\![1, N]\!], \quad \widehat{\Delta}_k^i = \mathcal{C}_{\rm up}\left(g_{k+1}^i - h_k^i\right) \\ \Omega_{k+1} = \mathcal{C}_{\rm dwn}\left(\frac{1}{N}\sum_{i=1}^N (\widehat{\Delta}_k^i + h_k^i)\right) \\ w_{k+1} = w_k - \gamma \Omega_{k+1} \end{array} \right. . \tag{2.2}$$

Artemis framework encompasses in particular four algorithms: the variant with unidirectional compression ($\omega_{\mathcal{C}}^{\rm dwn} = 0$) w.o. or with memory ($\alpha = 0$ or $\alpha \neq 0$) recovers QSGD defined by Alistarh et al. [4] and DIANA proposed by [81]. The variant using bidirectional

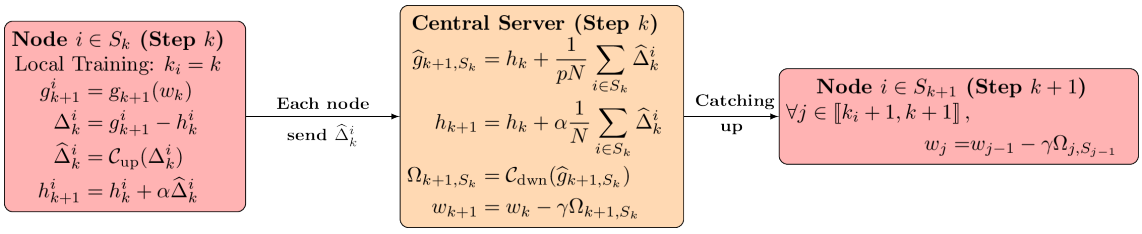


Figure 2.3: Summary of `Artemis`. At iteration $k \in \mathbb{N}$. w_k is the model's parameter, $\mathcal{C}_{\rm up}$ and $\mathcal{C}_{\rm dwn}$ are the compression operators, γ is the step size, α is the learning rate for the memory mechanism.

compression ($\omega_{\mathcal{C}}^{\rm dwn} \neq 0$) w.o memory ($\alpha = 0$) is called `Bi-QSGD`. The last and most effective variant combines bidirectional compression *with* memory and is the one we refer to as `Artemis` if no precision is given. A comparison of the algorithms is given in Table 2.1.

Assumptions. We consider assumptions A1, 3 and 7 to 9, introduced in Chapter 1. Assumption A7 differed from previously used assumptions in the literature. Unlike `Diana` [81, 72], `Dore` [74], `Dist-EF-SGD` [135], for `Double-Squeeze` [118], we assumed the variance of the noise to be bounded *only at optimal point* w_* and not *at any point* w in $\mathbb{R}^d$. When the variance is null ($\sigma_*^2 = 0$) at optimal point, we obtain a linear convergence while previous results obtain such convergence only if the variance is null *at any point* (i.e. only for deterministic GD). We hereafter state the general guarantee on `Artemis`.

Theorem 2.1 (Convergence of `Artemis`)**.** *Under A1, 3 and 7 to 9, there exists a* $\gamma_{\max}$ *such that for any step size* $\gamma \leqslant \gamma_{\max}$*, for a learning rate* $\alpha \in \{0, (\omega_{\mathcal{C}}^{\rm up}+1)^{-1}\}$ *and for any* k *in* $\mathbb{N}$*, the mean squared distance to* w_* *decreases at a linear rate up to a constant of the order of* E_α*:*

$$\mathbb{E}\left[\|w_k - w_*\|^2\right] \leqslant (1-\gamma\mu)^k\left(\delta_0^2 + (\omega_{\mathcal{C}}^{\rm up}+1)\gamma^2\tau_0\right) + \frac{2\gamma E_\alpha}{\mu N},$$

with, $E_0 = (\omega_{\mathcal{C}}^{\rm dwn}+1)\left((\omega_{\mathcal{C}}^{\rm up}+1)\frac{\sigma_*^2}{b} + \omega_{\mathcal{C}}^{\rm up}B^2\right)$ *and* $E_{(\omega_{\mathcal{C}}^{\rm up}+1)^{-1}} = \frac{\sigma_*^2}{b}(2\omega_{\mathcal{C}}^{\rm up}+1)(\omega_{\mathcal{C}}^{\rm dwn}+1)$*. The constant* τ_0 *depends on* $\sum_{i=1}^N \|h_0^i - h_*^i\|^2$*, and can be made independent of* B^2*.*

We can make the following remarks:

1. **Linear convergence.** The convergence rate given in Theorem 2.1 can be decomposed into two terms: a bias term, forgotten at linear speed $(1-\gamma\mu)^k$, and a variance residual term which corresponds to the *saturation level* of the algorithm. The rate of convergence $(1-\gamma\mu)$ does not depend on the variant of the algorithm. However, the variance and initial bias do vary.
2. **Variance term and memory.** The variance depends a) on both σ_*^2/b, and the distributions' difference B^2 without memory b) only on the gradients' variance *at the optimum* σ_*^2/b with memory. Similar theorems in related literature [74, 4, 81, 132, 118, 135] systematically had a worse bound for the variance term depending on a *uniform bound of the noise variance* or under much stronger conditions on the compression operator. [P1] and [74] were the firsts to give a linear convergence up to a threshold for bidirectional compression.
3. **Impact of memory.** This work was one the first ones on double compression that explicitly tackled the non i.i.d. case. We prove that memory makes the saturation threshold independent of B^2 for `Artemis`.

Table 2.1: Comparison of frameworks for main algorithms handling (bidirectional) compression. References: see [4] for QSGD, [81] for Diana, [51] for [HR20], [74] for Dore, [C7] for MCM and [118] for DoubleSqueeze

	QSGD	Diana	[HR20]	Dore	Double Squeeze	Dist EF-SGD	MCM	Artemis (new)
Data	i.i.d.	non i.i.d.	non i.i.d.	non i.i.d.	i.i.d.	i.i.d.	non i.i.d.	non i.i.d.
Bounded variance	Uniformly	Uniformly	Uniformly	Uniformly	Uniformly	Uniformly	Uniformly	At optimal point
Compression	One-way	One-way	One-way	Two-way	Two-way	Two-way	Two-way	Two-way
Error-feedback			✓	✓	✓	✓		
Memory		✓		✓			✓	✓
Device sampling			✓				✓	✓

4. **Variance term.** The variance term increases with a factor proportional to $\omega_{\mathcal{C}}^{\mathrm{up}}$ for the unidirectional compression, and proportional to $\omega_{\mathcal{C}}^{\mathrm{up}} \times \omega_{\mathcal{C}}^{\mathrm{dwn}}$ for bidirectional. This is the counterpart of compression, each compression resulting in a multiplicative factor on the noise. A similar increase in the variance appears in [81] and [74]. The noise level is attenuated by the number of devices N, to which it is inversely proportional.
5. **Link with classical SGD.** For variant of Artemis with $\alpha = 0$, if $\omega_{\mathcal{C}}^{\mathrm{up/dwn}} = 0$ (i.e. no compression) we recover SGD results: convergence does not depend on B^2, but only on the noise's variance.

Overall, it appears that Artemis is able to efficiently accelerate the learning during first iterations, enjoying the same linear rate as SGD with lower communication complexity, but it saturates at a higher level, proportional to σ_*^2 and independent of B^2.

Extensions: in [P1], we provide a convergence rate for Polyak-Ruppert averaging without strong convexity, resulting from Theorem 2.1, with a sequence of step size decaying to 0 with the horizon. A detailed discussion onthe maximal learning rate and constants is also provided. These results are omitted here for the sake of brevity.

Convergence in distribution and lower bound To prove that the increase in the variance in Theorem 2.1 is not an artifact of the proof, we prove the existence of a limit distribution for the iterates of Artemis, and analyze its variance. More precisely, we show a linear rate of convergence for the distribution Θ_k of w_k (launched from w_0), w.r.t. the Wasserstein distance $\mathcal{W}_2$ [126]: this gives us a lower bound on the asymptotic variance. Here, we further assume that the compression operator is *random sparsification* [128].

Theorem 2.2 (Convergence in distribution and lower bound on the variance)**.** *Under A1, 3 and 7 to 9, for $\gamma \leqslant \gamma_{\max}, \alpha \in \{0, (\omega_{\mathcal{C}}^{\mathrm{up}} + 1)^{-1}\}$ and:*

1. *There exists a limit distribution $\pi_{\gamma,v}$ depending on the variant v of the algorithm, s.t. for any $k \geqslant 1$, $\mathcal{W}_2(\Theta_k, \pi_{\gamma,v}) \leqslant (1-\gamma\mu)^k C_0$, with C_0 a constant.*
2. *When k goes to ∞, the second order moment $\mathbb{E}[\|w_k - w_*\|^2]$ converges to $\mathbb{E}_{w\sim\pi_{\gamma,v}}[\|w - w_*\|^2]$, which is* lower bounded *by $\Omega(\gamma E_\alpha/\mu N)$ as $\gamma \to 0$ with E_α as in Theorem 2.1.*

Interpretation. The second point (2.) means that the upper bound on the saturation level provided in theorem 2.1 *is tight* w.r.t. $\sigma_*^2, \omega_{\mathcal{C}}^{\mathrm{up}}, \omega_{\mathcal{C}}^{\mathrm{dwn}}, B^2, N$ and γ. Especially, it proves that there is indeed a quadratic increase in the variance w.r.t. $\omega_{\mathcal{C}}^{\mathrm{up}}$ and $\omega_{\mathcal{C}}^{\mathrm{dwn}}$ when using bidirectional compression. Altogether, these three theorems prove that bidirectional

compression can become strictly worse than usual stochastic gradient descent in high precision regimes.

Proof and assumptions. This theorem also naturally requires, for the second point, A1, 7 and 9 to be "tight": that is, e.g. $\mathrm{Var}(g^i_{k+1,*}) \geqslant \Omega(\sigma_*^2/b)$.

Extension: in [P1], we provide a detailed discussion on partial participation and how to tackle it.

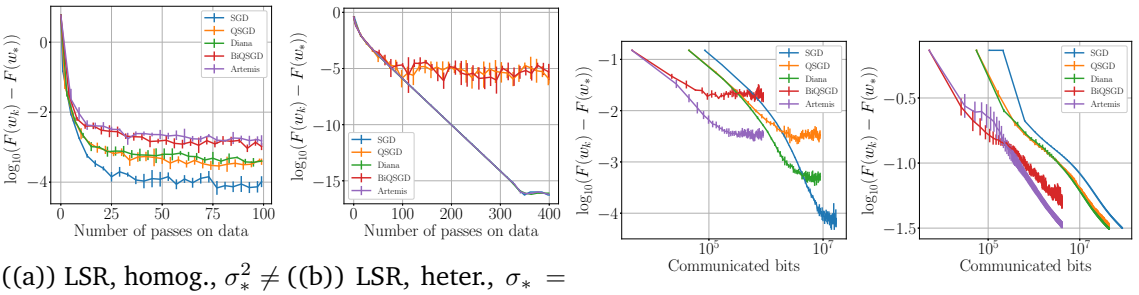
Experiments Numerical experiments confirm the theoretical findings in Theorem 2.1, and highlight the impact of the memory. We focus on five of the algorithms covered by our framework: `Artemis` with bidirectional compression (simply denoted `Artemis`), `QSGD`, `Diana`, `Bi-QSGD`, and usual `SGD` without any compression.

We here report the results on two synthetic and two real-world dataset: *superconduct* [see 49, with 21 263 points and 81 features] and *quantum* [see 12, with 50 000 points and 65 features] with $N = 20$ workers. They correspond to respectively logistic regression (LR) and least-squares regression (LSR). We simulate non-i.i.d. *and* unbalanced workers, by splitting the dataset in heterogeneous groups More experiments and details are given in [P1]. In all experiments, we display the logarithm excess error $\log_{10}(F(w_k) - F(w_*))$ w.r.t. the number of iterations k or the number of communicated bits. We use a quantization scheme (defined in Section 1.2) with $s = 2^0$ in full participation settings. Curves are averaged over 5 runs, and we plot error bars on all figures.

Convergence and complexity. Figure 2.4(a) presents the convergence of each algorithm w.r.t. the number of iterations k. During first iterations all algorithms make fast progress. However, because $\sigma_*^2 \neq 0$, all algorithms saturate; and the saturation level is higher for double compression (`Artemis`, `Bi-QSGD`), than for simple compression (`Diana`, `QSGD`), or than for SGD. This corroborates findings in Theorem 2.1 and Theorem 2.2. Figure 2.4(b) illustrates the **linear convergence under *null variance at the optimum***. Saturation is observed in Figure 2.4(a), but not if we consider a situation in which $\sigma_*^2 = 0$, and where the uniform bound on the gradient's variance *is not null*. Experiments also confirm that the control variate **is only beneficial with heterogeneity**: while in Figure 2.4(a), data is i.i.d. on machines, and `Artemis` is thus not expected to outperform `Bi-QSGD` (the difference between the two being the $\alpha > 0$), in Figures 2.4(c) and 2.4(d) we use *heterogeneous datasets*.

On Figures 2.4(c) and 2.4(d), the loss is plotted w.r.t. the theoretical number of bits exchanged after k iterations for the *quantum* and *superconduct* dataset. This confirms that double compression should be the method of choice to achieve a reasonable precision (*w.r.t.* σ_*), whereas for high precision, a simple method like SGD results *in a lower complexity*.

Conclusion We propose `Artemis`, a framework using bidirectional compression to reduce the number of bits needed to perform distributed or federated learning. On top of compression, `Artemis` includes a memory mechanism which improves convergence over non-i.i.d. data. As PP is a classical setting, we designed an approach (**PP2**) to tackle it while leveraging the full impact of memory, outperforming existing solutions. We provide three tight theorems giving guarantees of a fast convergence (linear up to a threshold), highlighting the impact of memory, analyzing Polyak-Ruppert averaging and obtaining lowers bound by studying convergence in distribution of our algorithm. Altogether, this



((a)) LSR, homog., $\sigma_*^2 \neq 0$ ((b)) LSR, heter., $\sigma_* = 0$. ((c)) Quantum ((d)) Superconduct

Figure 2.4: (a)&(b): synthetic datasets (a) illustration of the saturation when $\sigma_* \neq 0$ and data is i.i.d., (b) illustration of the memory benefits when $\sigma_* = 0$, with non-i.i.d. data. (c)&(d) Real datasets, with heterogeneity, $\sigma_* \neq 0$, $N = 20$ workers, $p = 1$, $b > 1$ (150 iter.). X-axis in # bits.

improves the understanding of compression combined with a memory mechanism and sheds light on challenges ahead.

Extension. We refer to [P1] for more experiments, proofs, and a discussion the bi-directional compression framework.

2.3 Federated Expectation Maximization with compression

Summary of [C6]: The Expectation Maximization (EM) algorithm is the default algorithm for inference in latent variable models. As in any other field of machine learning, applications of latent variable models to very large datasets makes the use of advanced parallel and distributed architectures mandatory. In [C6] we introduced `FedEM`, which was one of the first extension of the EM algorithm to the federated learning context. `FedEM` is a communication efficient method, which handles partial participation of local devices, and is robust to heterogeneous distributions of the datasets. To alleviate the communication bottleneck, `FedEM` compresses appropriately defined complete data sufficient statistics. We also developed and analyzed an extension of `FedEM` to further incorporate a variance reduction scheme. In all cases, we derived finite-time complexity bounds for smooth non-convex problems. Numerical results are presented to support our theoretical findings, as well as an application to federated missing values imputation for biodiversity monitoring.

2.3.1 Introduction to EM algorithm.

The Expectation Maximization (EM) algorithm is the most popular approach for inference in latent variable models. The EM algorithm, a special instance of the Majorize/Minimize algorithm [66], was formalized by [19] and is without doubt one of the fundamental algorithms in machine learning. Applications include among many others finite mixture analysis, latent factor models inference, and missing data imputation; see [83, 79, 33] and the references therein. As in any other field of machine learning, training latent variable models on very large datasets make the use of advanced parallel and distributed architectures mandatory.

The conventional EM algorithm is not suitable for FL settings. We propose several new distributed versions of the EM algorithm supporting compressed communication. More precisely, our objective is to minimize a non-convex finite-sum smooth objective function

$$\arg\min_{w\in\mathcal{W}} F(w), \qquad F(w) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i(w) + R(w)\ , \qquad \mathcal{W} \subseteq \mathbb{R}^d\ , \tag{2.3}$$

where N is still the number of workers which are connected to the central server, and the worker $\#i$ only has access to its local data; finally R is a penalty term which may be introduced to promote sparsity, regularity, etc. In latent variable models, $\mathcal{L}_i(w) = -m^{-1}\sum_{j=1}^{m} \log p(y_{ij}; w)$, where $\{y_{ij}\}_{j=1}^{m}$ are the m observations available for worker $\#i$, and $p(y; w)$ is the *incomplete* likelihood. $p(y; w)$ is defined by marginalizing the *complete-data* likelihood $p(y, z; w)$ defined as the joint probability density function of the observation y and a non-observed latent variable $z \in \mathsf{Z}$, i.e. $p(y; w) = \int_{\mathsf{Z}} p(y, z; w)\mu(\mathrm{d}z)$ where Z is the *latent space* and μ is a measure on Z. We focus in this section on the case where $p(y, z; w)$ belongs to a curved exponential family, given by

$$p(y, z; w) \stackrel{\text{def}}{=} \rho(y, z) \exp\left\{ \langle s(y, z), \phi(w)\rangle - \psi(w)\right\}\ ; \tag{2.4}$$

where $s(y, z) \in \mathbb{R}^q$ is the *complete-data sufficient statistics*, $\phi : \mathcal{W} \to \mathbb{R}^q$ and $\psi : \mathcal{W} \to \mathbb{R}$, $\rho : \mathsf{Y} \times \mathsf{Z} \to \mathbb{R}^+$ are vector/scalar functions.

In absence of communication constraints, the EM algorithm is a popular method to solve (2.3). It alternates between two steps: in the Expectation (E) step, using the current value of the iterate w_{curr}, it computes a majorizing function $w \mapsto \mathsf{Q}(w, w_{\text{curr}})$ given up to an additive constant by

$$\mathsf{Q}(w, w_{\text{curr}}) \stackrel{\text{def}}{=} -\langle \bar{\mathsf{s}}(w_{\text{curr}}), \phi(w)\rangle + \psi(w) + R(w) \quad \text{where} \quad \bar{\mathsf{s}}(w) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} \bar{\mathsf{s}}_i(w)\ ; \tag{2.5}$$

and $\bar{\mathsf{s}}_i(w)$ is the ith device conditional expectation of the complete-data sufficient statistics:

$$\bar{\mathsf{s}}_i(w) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{j=1}^{m} \bar{\mathsf{s}}_{ij}(w)\ , \quad \bar{\mathsf{s}}_{ij}(w) \stackrel{\text{def}}{=} \int_{\mathsf{Z}} s(y_{ij}, z) p(z|y_{ij}; w)\mu(\mathrm{d}z)\ , \tag{2.6}$$

where $p(z|y_{ij}; w) \stackrel{\text{def}}{=} p(y_{ij}, z; w)/p(y_{ij}; w)$. As for the M step, an updated value of w_{curr} is computed as a minimizer of $w \mapsto \mathsf{Q}(w, w_{\text{curr}})$. The majorizing function is then updated with the new w_{curr}; this process is iterated until convergence. The EM algorithm is most useful when for any $w_{\text{curr}} \in \mathcal{W}$, the function $w \mapsto \mathsf{Q}(w, w_{\text{curr}})$ is a convex function of the parameter w which is solvable in w either explicitly or with little computational effort. A major advantage of the EM algorithm stems from its invariance under homeomorphisms, contrary to classical first-order methods: the EM updates are the same for any continuous invertible re-parametrization [65].

Challenges in FL. In the FL context, the vanilla EM algorithm is affected by two major problems: the communication bottleneck and data heterogeneity.

1. **Compression.** As in the previous section, we use *communication compression*, but contrary to Section 2.2, compression is not applied to stochastic gradients.
2. **Heterogeneity.** Our model in Equations (2.3), (2.5) and (2.6) allows the local loss functions to depend on the worker $i \in \{1, \dots, N\}$ and the observations y_{ij} to be independent but not necessarily identically distributed. In addition, our theoretical results deal with specific behaviors for each worker $i \in \{1, \dots, N\}$, see e.g., A15

and 16. In the FL-EM setting, heterogeneity manifests itself by the non-equality of the *local* conditional expectations of the complete-data sufficient statistics $\bar{\mathsf{s}}_i$'s; modifications to the algorithms must be performed to ensure convergence at the central server.

Contributions:

- **`FedEM`.** The main highlighted contribution in this section is a new method called `FedEM`, supporting communication compression, partial participation and data heterogeneity. In this algorithm, workers compute an estimate of the *local complete-data sufficient statistics* $\bar{\mathsf{s}}_i$ using a minibatch of data, apply an unbiased compression operator to a *noise compensated version* (using the control variate technique highlighted in this chapter) and send the result to the central server, which performs aggregation and the M-step (i.e., parameter update).
- **Theoretical analysis.** EM in the curved exponential family setting converges to the roots of a function `h`. We introduce a unified theoretical framework which covers the convergence of `FedEM` and `VR-FedEM` algorithms in the non-convex case and establish convergence guarantees for finding an ε-stationary point (see Theorem 2.3). We provide the number $K_{\rm opt}(\varepsilon)$ of optimization steps and the number $K_{\rm CE}(\varepsilon)$ of computed conditional expectations to reach ε-stationarity. This sheds light on the tradeoffs of compression and the robustness of `FedEM` to data heterogeneity.

Extensions: In [C6], we also propose to improve `FedEM` by adding a variance reduction on the individual datapoints, inspired by the `SPIDER` framework [29] which has recently been extended to the EM framework [31]. `VR-FedEM` does not require the step sizes to decrease with m and achieves state of the art iteration complexity to reach a precision ε, while being robust to heterogeneity.

2.3.2 `FedEM`: Expectation Maximization algorithms for FL

Recall the definition of the negative penalized (normalized) log-likelihood $F(w)$ from (2.3). Along the entire section, we make the following assumptions A11 to A13:

A11. *The parameter set $\mathcal{W} \subseteq \mathbb{R}^d$ is a convex open set. The functions $R : \mathcal{W} \to \mathbb{R}$, $\phi : \mathcal{W} \to \mathbb{R}^q$, $\psi : \mathcal{W} \to \mathbb{R}$, and $\rho(y_{ij}, \cdot) : \mathsf{Z} \to \mathbb{R}_+$, $s(y_{ij}, \cdot) : \mathsf{Z} \to \mathbb{R}^q$ for $i \in [N]^\star$ and $j \in [m]^\star$ are measurable functions. For any $w \in \mathcal{W}$ and $i \in [N]^\star$, the log-likelihood is finite: $-\infty < \mathcal{L}_i(w) < \infty$.*

A12. *For all $w \in \mathcal{W}$ and $i \in [N]^\star$, the conditional expectation $\bar{\mathsf{s}}_i(w)$ is well-defined.*

A13. *For any $s \in \mathbb{R}^q$, the map $s \mapsto \arg\min_{w \in \mathcal{W}} \{\psi(w) + R(w) - \langle s, \phi(w)\rangle\}$ exists and is unique; the singleton is denoted by $\{\mathsf{T}(s)\}$.*

EM defines a sequence $\{w_k, k \geqslant 0\}$ that can be computed recursively as $w_{k+1} = \mathsf{T} \circ \bar{\mathsf{s}}(w_k)$, where the map T is defined in A13 and $\bar{\mathsf{s}}$ is defined in (2.5). On the other hand, the EM algorithm can be defined through a mapping in the complete-data sufficient statistics, referred to as the *expectation space*. In this setting, the EM iteration defines a $\mathbb{R}^q$-valued sequence $\{\widehat{S}_k, k \geqslant 0\}$ given by $\widehat{S}_{k+1} = \bar{\mathsf{s}} \circ \mathsf{T}(\widehat{S}_k)$. Thus, we observe that the EM algorithm admits two equivalent representations:

$$\text{(Parameter space)} \quad w_{k+1} = \mathsf{T} \circ \bar{\mathsf{s}}(w_k); \quad \text{(Expectation space)} \quad \widehat{S}_{k+1} = \bar{\mathsf{s}} \circ \mathsf{T}(\widehat{S}_k). \tag{2.7}$$

In this section, we focus on the *expectation space representation*; see [65] for an interesting discussion on the connection of EM and mirror descent. It has been shown in [18] that if $s_\star$ is a fixed point to the EM algorithm in the expectation space, then $w_\star \overset{\text{def}}{=} \mathsf{T}(s_\star)$ is a fixed point of the EM algorithm in the parameter space, i.e., $w_\star = \mathsf{T} \circ \bar{\mathsf{s}}(w_\star)$; note that the converse is also true. Define the functions h_i and h from $\mathbb{R}^q$ to $\mathbb{R}^q$ by $\mathsf{h}(s) \overset{\text{def}}{=} \frac{1}{N}\sum_{i=1}^N \mathsf{h}_i(s)$ with $\mathsf{h}_i(s) \overset{\text{def}}{=} \bar{\mathsf{s}}_i \circ \mathsf{T}(s) - s$.

$$\mathsf{h}(s) \overset{\text{def}}{=} \frac{1}{N}\sum_{i=1}^N \mathsf{h}_i(s) \;, \qquad \mathsf{h}_i(s) \overset{\text{def}}{=} \bar{\mathsf{s}}_i \circ \mathsf{T}(s) - s \;. \tag{2.8}$$

A key property is that the fixed points of EM in the expectation space are the roots of the *mean field* $s \mapsto \mathsf{h}(s)$ (see (2.5) for the definition of $\bar{\mathsf{s}}$). Therefore, convergence of EM-based algorithms is evaluated in terms of ε-**stationarity** (see [36, 31]): for all $\varepsilon > 0$, there exists a (possibly random) termination time K s.t.: $\mathbb{E}\left[\|\mathsf{h}(\widehat{S}_K)\|^2\right] \leqslant \varepsilon$. Another key property of EM is that it is a monotonic algorithm: each iteration leads to a decrease of the negative penalized log-likelihood i.e. $F(w_{k+1}) \leqslant F(w_k)$ or, equivalently in the expectation space $F \circ \mathsf{T}(\widehat{S}_{k+1}) \leqslant F \circ \mathsf{T}(\widehat{S}_k)$ (for sequences $\{w_k, k \geqslant 0\}$ and $\{\widehat{S}_k, k \geqslant 0\}$ given by (2.7)). A 14 assumes that the roots of the mean field h are the roots of the gradient of $F \circ \mathsf{T}$ (see [18] for the same assumption when studying Stochastic EM). A15 assumes global Lipschitz properties of the functions h_i's.

A14. *The function* $\mathrm{W} \overset{\text{def}}{=} F \circ \mathsf{T} : \mathbb{R}^q \to \mathbb{R}$ *is continuously differentiable on* $\mathbb{R}^q$ *and its gradient is globally Lipschitz with constant* $L_{\dot{\mathrm{W}}}$. *Furthermore, for any* $s \in \mathbb{R}^q$, $\nabla \mathrm{W}(s) = -B(s)\mathsf{h}(s)$ *where* $B(s)$ *is a* $q \times q$ *positive definite matrix. In addition, there exist* $0 < v_{\min} \leqslant v_{\max}$ *such that for any* $s \in \mathbb{R}^q$, *the spectrum of* $B(s)$ *is in* $[v_{\min}, v_{\max}]$.

A15. *For any* $i \in [N]^\star$, *there exists* $L_i > 0$ *such that for any* $s, s' \in \mathbb{R}^q$, $\|\mathsf{h}_i(s) - \mathsf{h}_i(s')\| = \|(\bar{\mathsf{s}}_i \circ \mathsf{T}(s) - s) - (\bar{\mathsf{s}}_i \circ \mathsf{T}(s') - s')\| \leqslant L_i \|s - s'\|$.

Finally, we recall that we assume A1 on the compression operator. As the compression is only performed in the uplink direction, we denote $\omega := \omega_{\mathrm{up}}$.

Schematic visualization of a naive Federated EM algorithm. On Figure 2.5, we provide a schematic representation of a simplified version of `FedEM`. The left part represents the *expectation space*, the right part *parameter space*. In short, we alternate between stochastic approximation steps to estimate the sufficient statistic, based on each workers $i \in [N]$ oracle, aggregation steps, and optimization steps (computing T). For visualization purpose, the compression step is omitted on the figure, but importantly, the memory term $V_{k,i}$ and the compression are applied onto the *statistics* space (left space), before the communication to the central server and the aggregation step.

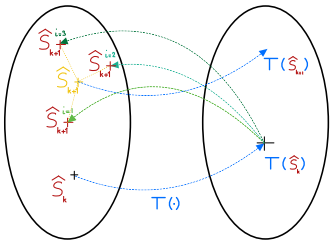


Figure 2.5: Schematic visualization on a naive Federated EM

Algorithm 1: `FedEM` with partial participation

Data: $k_{\max} \in \mathbb{N}^\star$; for $i \in [N]^\star$, $V_{0,i} \in \mathbb{R}^q$; $\widehat{S}_0 \in \mathbb{R}^q$; a positive sequence $\{\gamma_{k+1}, k \in [k_{\max}-1]\}$; $\alpha > 0$; a coefficient $p = \mathbb{E}_{\mathcal{A}\sim\mathbb{P}_{\rm PP}}[\mathrm{card}(\mathcal{A})]/N$.

Result: The `FedEM-PP` sequence: $\{\widehat{S}_k, k \in [k_{\max}]\}$

1 Set $V_0 = N^{-1}\sum_{i=1}^N V_{0,i}$
2 **for** $k = 0, \ldots, k_{\max}-1$ **do**
3 Sample $\mathcal{A}_{k+1} \sim \mathbb{P}_{\rm PP}$
4 **for** $i \in \mathcal{A}_{k+1}$ **do**
5 *(worker #i)*
6 Sample $\mathsf{S}_{k+1,i}$, an approximation of $\bar{\mathsf{s}}_i \circ \mathsf{T}(\widehat{S}_k)$
7 Set $\Delta_{k+1,i} = \mathsf{S}_{k+1,i} - V_{k,i} - \widehat{S}_k$
8 Set $V_{k+1,i} = V_{k,i} + \alpha\, \mathcal{C}(\Delta_{k+1,i})$.
9 Send $\mathcal{C}(\Delta_{k+1,i})$ to the central server
10 **for** $i \notin \mathcal{A}_{k+1}$ **do**
11 *(worker #i)*
12 Set $V_{k+1,i} = V_{k,i}$ (no update)
13 *(the central server)*
14 Set $H_{k+1} = V_k + (Np)^{-1}\sum_{i\in\mathcal{A}_{k+1}} \mathcal{C}(\Delta_{k+1,i})$
15 Set $\widehat{S}_{k+1} = \widehat{S}_k + \gamma_{k+1} H_{k+1}$
16 Set $V_{k+1} = V_k + \alpha N^{-1}\sum_{i\in\mathcal{A}_{k+1}} \mathcal{C}(\Delta_{k+1,i})$
17 Send $\widehat{S}_{k+1}$ and $\mathsf{T}(\widehat{S}_{k+1})$ to the N workers

The `FedEM` algorithm , is described by algorithm 1. The algorithm encompasses partial participation of the workers: at iteration $\#(k+1)$, only a subset $\mathcal{A}_{k+1}$ of active workers participate to the training, see line 3. The averaged fraction of participating workers is denoted p. Each of the active workers $\#i$ computes an *unbiased* approximation $\mathsf{S}_{k+1,i}$ (line 6) of $\bar{\mathsf{s}}_i \circ \mathsf{T}(\widehat{S}_k)$; conditionally to the past, these approximations are independent. The workers then transmit to the central server a compressed information about the new sufficient statistics.

Control variates $(V_{k,i})_{i\in[N]}$**.** A naive solution would be to compress and transmit $\mathsf{S}_{k+1,i} - \widehat{S}_k$, but data heterogeneity between servers often prevents these local differences from vanishing at the optimum, leading to large compression errors and impairing convergence of the algorithm (as described in Section 2.1). Following the approach highlighted in this chapter, a memory $V_{k,i}$ (initialized to $\mathsf{h}_i(\widehat{S}_0)$ at $k=0$) is introduced; and the *differences* $\Delta_{k+1,i} \overset{\text{def}}{=} \mathsf{S}_{k+1,i} - \hat{S}_k - V_{k,i}$ are compressed for $i \in \mathcal{A}_{k+1}$ (line 7 and line 9). These memories are updated locally: $V_{k+1,i} = V_{k,i} + \alpha\,\mathcal{C}(\Delta_{k+1,i})$, at line 8, with $\alpha > 0$ (typically set to $1/(1+\omega)$ with ω the compression constant). On its side, the central server releases an aggregated estimate $\widehat{S}_{k+1}$ of the complete-data sufficient statistics by averaging the quantized difference $(np)^{-1}\sum_{i\in\mathcal{A}_{k+1}} \mathcal{C}(\Delta_{k+1,i})$ and by adding V_k (line 14 and line 15). Then, it updates $V_{k+1} = V_k + \alpha n^{-1}\sum_{i=1}^N \mathcal{C}(\Delta_{k+1,i})$, see line 16. The final step consists in solving the M-step of the EM algorithm, i.e. in computing $\mathsf{T}(\widehat{S}_{k+1})$ (see A13).

The convergence analysis is under the following assumptions on the oracle $\mathsf{S}_{k+1,i}$:

for any $i \in [n]^\star$, the approximations $\mathsf{S}_{k+1,i}$ are unbiased and their conditional variances are uniformly bounded in k. For each $k \in \mathbb{N}$, denote by $\mathcal{F}_k$ the σ-algebra generated by $\{\mathsf{S}_{\ell,i}, \mathcal{A}_\ell; i \in [n]^\star, \ell \in [k]\}$ and including the randomness inherited from the quantization operator $\mathcal{C}$ up to iteration $\#k$.

A16. *For all $k \in \mathbb{N}$, conditional to $\mathcal{F}_k$, $\{\mathsf{S}_{k+1,i}\}_{i=1}^N$ are independent. Moreover, for any $i \in [N]^\star$, $\mathbb{E}\left[\mathsf{S}_{k+1,i}|\mathcal{F}_k\right] = \bar{\mathsf{s}}_i \circ \mathsf{T}(\widehat{S}_k)$ and there exists $\sigma_i^2 > 0$ such that for any $k \geqslant 0$*

$$\mathbb{E}\left[\left\|\mathsf{S}_{k+1,i} - \bar{\mathsf{s}}_i \circ \mathsf{T}(\widehat{S}_k)\right\|^2 \middle| \mathcal{F}_k\right] \leqslant \sigma_i^2 .$$

A16 covers both the finite-sum setting described in the introduction, and the online setting. In the finite-sum setting, $\bar{\mathsf{s}}_i$ is of the form $m^{-1}\sum_{j=1}^m \bar{\mathsf{s}}_{ij}$. In that case, $\mathsf{S}_{k+1,i}$ can be the sum over a minibatch $\mathcal{B}_{k+1,i}$ of size b sampled at random in $[m]^\star$, with or without replacement and independently of the history of the algorithm: we have $\mathsf{S}_{k+1,i} = \mathsf{b}^{-1}\sum_{j\in\mathcal{B}_{k+1,i}} \bar{\mathsf{s}}_{ij} \circ \mathsf{T}(\widehat{S}_k)$. In the online setting, the oracles $\mathsf{S}_{k+1,i}$ come from an online processing of streaming informations; in that case $\mathsf{S}_{k+1,i}$ can be computed from a minibatch of independent examples so that the conditional variance σ_i^2, which will be inversely proportional to the size of the minibatch, can be made arbitrarily small.

Convergence analysis. We now present in Theorem 2.3 our key result, from which complexity expressions are derived.

Theorem 2.3. *Assume A11 to A16 and set $L^2 \overset{def}{=} n^{-1}\sum_{i=1}^N L_i^2$, $\sigma^2 \overset{def}{=} N^{-1}\sum_{i=1}^N \sigma_i^2$. Let $\{\widehat{S}_k, k \in [k_{\max}]\}$ be given by algorithm 1, with $\omega > 0$, $\alpha \overset{def}{=} (1+\omega)^{-1}$ and $\gamma_k = \gamma \in (0, \gamma_{\max}]$ where*

$$\gamma_{\max} \overset{def}{=} \frac{v_{\min}}{2L_{\dot{\mathrm{W}}}} \wedge \frac{\sqrt{N}}{2\sqrt{2}L(1+\omega)\sqrt{\omega}} . \tag{2.9}$$

Denote by K the uniform random variable on $[k_{\max}-1]$. Then, taking $V_{0,i} = \mathsf{h}_i(\widehat{S}_0)$ for all $i \in [N]^\star$:

$$v_{\min}\left(1 - \gamma\frac{L_{\dot{\mathrm{W}}}}{v_{\min}}\right)\mathbb{E}\left[\left\|\mathsf{h}(\widehat{S}_K)\right\|^2\right] \leqslant \frac{1}{\gamma k_{\max}}\left(\mathrm{W}(\widehat{S}_0) - \min \mathrm{W}\right) + \gamma L_{\dot{\mathrm{W}}}\frac{1+5\omega}{N}\sigma^2 . \tag{2.10}$$

When there is no compression ($\omega = 0$ so that $\mathcal{C}(s) = s$), we prove that the introduction of the random variables $V_{k,i}$'s play no role whatever $\alpha > 0$ and the choice of the $V_{0,i}$'s, and we have for any $\gamma \in (0, 2v_{\min}/L_{\dot{\mathrm{W}}})$ in the supplemental)

$$\left(1 - \gamma\frac{L_{\dot{\mathrm{W}}}}{2v_{\min}}\right)\mathbb{E}\left[\left\|\mathsf{h}(\widehat{S}_K)\right\|^2\right] \leqslant \frac{1}{\gamma k_{\max}}\left(\mathrm{W}(\widehat{S}_0) - \min \mathrm{W}\right) + \gamma L_{\dot{\mathrm{W}}}\frac{\sigma^2}{N} . \tag{2.11}$$

Optimizing the learning rate γ, we derive the following corollary.

Corollary 2.4 (of Theorem 2.3)**.** *Choose $\gamma \overset{def}{=} \left(\frac{(\mathrm{W}(\widehat{S}_0) - \min \mathrm{W})n}{k_{\max}L_{\dot{\mathrm{W}}}(1+5\omega)\sigma^2}\right)^{1/2} \wedge \gamma_{\max}$. We get*

$$\mathbb{E}\left[\left\|\mathsf{h}(\widehat{S}_K)\right\|^2\right] \leqslant \frac{4}{v_{\min}}\left(\sqrt{\frac{(\mathrm{W}(\widehat{S}_0) - \min \mathrm{W})L_{\dot{\mathrm{W}}}(1+5\omega)\sigma^2}{nk_{\max}}} \vee \frac{(\mathrm{W}(\widehat{S}_0) - \min \mathrm{W})}{\gamma_{\max}k_{\max}}\right) .$$

Theorem 2.3 and Corollary 2.4 do not require any assumption regarding the distributional heterogeneity of workers. These results remain thus valid when workers have access to data resulting from different distributions a widespread situation in FL frameworks. Crucially, without assumptions on the heterogeneity of workers, the convergence of a

"naive" implementation of compressed distributed EM (i.e. an implementation without the variables $V_{k,i}$'s) would not converge.

Let us comment the complexity to reach an ε-stationary point, and more precisely how the complexity evaluated in terms of the number of optimization steps depend on ω, N, σ^2 and ε. Since $\mathcal{K}_{\mathrm{Opt}}(\varepsilon) = k_{\max}$, from Corollary 2.4 we have that: $\mathcal{K}_{\mathrm{opt}}(\varepsilon) = O\Big(\frac{(1+\omega)\sigma^2}{n\varepsilon^2}\Big) \vee O\Big(\frac{1}{\gamma_{\max}\varepsilon}\Big)$.

Maximal learning rate and compression. The comparison of Theorem 2.3 with the no compression case (see (2.11)) shows that compression impacts $\gamma_{\max}$ by a factor proportional to $\sqrt{N}/\omega^{3/2}$ as ω increases (similar constraints were observed in the risk optimization literature, e.g. in [52, 96]). This highlights two different regimes depending on the ratio $\sqrt{N}/\omega^{3/2}$: if the number of workers N scales at least as ω^3, the maximal learning rate is not impacted by compression; on the other hand, for smaller numbers of workers $n \ll \omega^3$, compression can degrade the maximal learning rate. We highlight this conclusion with a small example in the case of scalar quantization for which $\omega \sim \sqrt{q}/s_{\mathrm{quant}}$: for $q = 10^2$ and $s_{\mathrm{quant}} = 4$ (obtaining a compression rate of a factor 16), the maximal learning rate is almost unchanged if $N \geqslant 16$.

Dependency on ε. The complexity $\mathcal{K}_{\mathrm{opt}}(\varepsilon)$ is decomposed into two terms scaling respectively as $\sigma^2\varepsilon^{-2}$ and $\gamma_{\max}^{-1}\varepsilon^{-1}$, the first term being dominant when $\varepsilon \to 0$. This observation highlights two different regimes: a *high noise regime* corresponding to $\gamma_{\max}(1+\omega)\sigma^2/(N\varepsilon^{-1}) \geq 1$ where the complexity is of order $\sigma^2\varepsilon^{-2}$, and a *low noise regime* where $\gamma_{\max}(1+\omega)\sigma^2/(N\varepsilon^{-1}) \leqslant 1$ and the complexity is of order $\gamma_{\max}^{-1}\varepsilon^{-1}$. An extreme example of the low noise case is $\sigma^2 = 0$, occurring for example in the finite-sum case (i.e., when $\bar{\mathsf{s}}_i = m^{-1}\sum_{j=1}^m \bar{\mathsf{s}}_{ij}$) with the oracle $\mathsf{S}_{k+1,i} = \bar{\mathsf{s}}_i \circ \mathsf{T}(\widehat{S}_k)$.

Impact of compression for ε-stationarity. As mentioned above, the compression simultaneously impacts the maximal learning rate (as in (2.9)) and the complexity $\mathcal{K}_{\mathrm{opt}}(\varepsilon)$. Consequently, the impact of the compression depends on the balance between ω, N, σ^2 and ε, and we can distinguish four different "main" regimes. In the following tabular, for each of the four situations, we summarize the *increase in complexity* $\mathcal{K}_{\mathrm{opt}}(\varepsilon)$ resulting from compression.

	Complexity regime: (Dominating term in $\mathcal{K}_{\mathrm{opt}}(\varepsilon)$)	$\frac{(1+\omega)\sigma^2}{n\varepsilon^2}$	$\frac{1}{\gamma_{\max}\varepsilon}$
$\gamma_{\max}$ regime: (Dominating term in (2.9))	Example situation	High noise σ^2, small ε	Low σ^2 (e.g., large minibatch) larger ε
$\frac{v_{\min}}{2L_{\dot{\mathrm{W}}}}$	large ratio N/ω^3	$\times\omega$	$\times 1$
$\frac{\sqrt{N}}{2\sqrt{2}L(1+\omega)\sqrt{\omega}}$	low ratio N/ω^3	$\times\omega$	$\times\omega^{3/2}/\sqrt{N}$

Depending on the situation, the complexity can be multiplied by a factor ranging from 1 to $\omega \vee (\omega^{3/2}/\sqrt{N})$. Remark that the communication cost of each iteration is typically reduced by compression of a factor at least ω. Moreover, the benefit of compression is most significant in the *low noise* regime and when the maximal learning rate is $v_{\min}/(2L_{\dot{\mathrm{W}}})$ (e.g., when N large enough). We then improve the communication cost of each iteration without increasing the optimization complexity, effectively reducing the communication budget "for free".

Extension and experiments. We refer to [C6] for the extension to `VR-FedEM` in which an *observation-wise* variance reduction scheme is also applied, and for the numerical results.

Conclusions We introduced `FedEM` which is, to the best of our knowledge, the first algorithm implementing EM in a FL setting, and handles compression of exchanged information, data heterogeneity and partial participation. We further extended it to incorporate a variance reduction scheme, yielding `VR-FedEM`. We derived complexity bounds which highlight the efficiency of the two algorithms, and illustrated our claims with numerical simulations, as well as an application to biodiversity monitoring data. In a simultaneously published work, Marfoq et al. [78] consider a different Federated EM algorithm, in order to address the personalization challenge by considering a mixture model. Under the assumption that each local data distribution is a mixture of unknown underlying distributions, their algorithm computes a model corresponding to each distribution. On the other hand, we focus on the curved exponential family, with variance reduction, partial participation and compression and on limiting the impact of heterogeneity, but do not address personalization.

Conclusion

This first chapter correspond to the first key idea in the domain of compression: (1) with compression, heterogeneity hinders convergence (2) it is possible to reduce the negative impact of heterogeneity relying on control variates. We gave two examples of such contributions. Similar ideas were also central in our work on Langevin dynamics [C8].

3

Model preservation and error compensation.

This chapter is based on [C7]: *Preserved central model for faster bidirectional compression in distributed settings*, C. Philippenko, **A.D.**, Neurips 2021.

The main contribution is to propose a method that takes advantage of the observation of *un-compressed* quantity, before they are compressed.

Intuition: model preservation and error feedback In Chapter 2, we showed how to recover the convergence rate of the homogeneous case with compression by relying on control variates, that learn the dissimilarity and modify the learning algorithm consequently. Nevertheless, compression still degrades the convergence of the algorithm: it amplifies the gradient noise by multiplicative factor. This is visible in Theorem 2.1: for `Artemis`, the limiting noise is amplified.

In this Chapter, we explore a different direction. The fundamental observation is that, when performing compression, the worker and the central server, who communicate a compressed version of an update, can also compute, store and eventually use the *difference* between the update and its compression. Such strategies, coined Error Feedback, were originally introduced to recover convergence for biased compression operators (e.g., signSGD, [58]). In [C7], we show that such a strategy is natural in the context of bi-directional compression for FL, and we introduce a new error compensation scheme based on *unbiased compression* and a memory scheme to control the variance of the process.

Summary of [C7]. We develop a new approach to tackle communication constraints in a distributed learning problem with a central server. We propose and analyze a new algorithm that performs bidirectional compression and achieves the same convergence rate as algorithms using only uplink (from the local workers to the central server) compression. To obtain this improvement, we design `MCM`, an algorithm such that the downlink compression *only impacts local models*, while the global model is preserved. As a result, and contrary to previous works, the gradients on local servers are computed on *perturbed models*. Consequently, convergence proofs are more challenging and require a precise control of this perturbation. To ensure it, `MCM` additionally combines model compression with a memory mechanism. This analysis opens new doors, e.g.

incorporating worker dependent randomized-models and partial participation.

Introduction As in Section 2.2 of Chapter 2, we consider the federated optimization problem Equation (1.1) with bidirectional compression. Existing bidirectional algorithms [118, 135, 104, 74, 95, 51, 131, 39] aggregate all the information they have received, compress it broadcast the result. Both the main "global" model and the "local" ones perform the *same* update with this compressed information. Consequently, the model hold on the central server and the one used on the local workers (to query the gradient oracle) are identical. However, this means that the model on the central server has been artificially *degraded*: instead of using all the information it has received, it is updated with the compressed information.

Here, we focus on *preserving* (instead of *degrading*) the central model: the update made on the server's side only weakly depends on the downlink compression. But simultaneously, the local models are *different* from the central model. The local gradients are thus measured on a *"perturbed model"* (or *"perturbed iterate"*): such an approach requires a more involved analysis and the algorithm must be carefully designed to control the deviation between the local and global models [77]. For example, algorithms directly compressing the model or the update would simply not converge.

We propose MCM - *Model Compression with Memory* - a new algorithm that 1) preserves the central model, and 2) uses a memory scheme to reduce the variance of the local model. We prove that the convergence of this method is similar to the one of algorithms using only unidirectional compression.

Contributions:

1. We propose a new algorithm MCM, combining a memory process to the "preserved" update. Gradients are observed at a random point, which, in expectation, is to the preserved model.
2. We carefully control the variance of the local models w.r.t. the global one. We provide a *contraction equation* involving the control on the local model's variance and show that MCM **achieves the same asymptotic rate of convergence as single compression in strongly-convex, convex and non-convex regimes**.
3. We propose a variant, Rand-MCM incorporating diversity into models shared with the local workers and show that it improves convergence for quadratic functions.

This is the first algorithm for double compression to focus on a **preserved central model**. We underline, both theoretically and in practice, that we get the same asymptotic convergence rate for simple and double compression - which is a major improvement. Our approach is one of the first to allow for worker dependent model, and to naturally adapt to worker dependent compression levels.

In the rest of this chapter, we present a few selected results form [C7]. We first introduce MCM, then provide one convergence rate in the convex case, that illustrates our claims.

3.1 Algorithm design: Preserved model and MCM algorithm

We consider the minimization problem Equation (1.1), and focus here on the convex case, assuming the existence of an optimal parameter w_*, and denoting $F_* = F(w_*)$.

As in Chapter 2, we rely on a stochastic gradient descent (SGD) algorithm. A stochastic gradient g_{k+1}^i is provided at iteration k in $\mathbb{N}$ to the device i in $[\![1, N]\!]$. This gradient oracle can be computed on a mini-batch of size b. We recall that we use two different compression operators, respectively $\mathcal{C}_{\text{up}}$ and $\mathcal{C}_{\text{dwn}}$ to compress messages exchanged in each direction.

In classical algorithms with double compression, including Artemis& Dore, the update is of the following form:

$$w_{k+1} = w_k - \gamma \mathcal{C}_{\text{dwn}}\left(\frac{1}{N}\sum_{i=1}^N \text{Update}_i(w_k) + \text{Correction}_i\right), \qquad \text{(Degraded Update)}$$

where Update_i and Correction_i vary depending on the algorithm, and are typically meant to tackle heterogeneity. For example, we can have $\text{Update}_i(w) = \mathcal{C}_{\text{up}}(g_{k+1}^i(w) - h_k^i)$ and $\text{Correction}_i = h_k^i$, as in Artemis. This approach has a major drawback: the central server receives and aggregates information $\frac{1}{N}\sum_{i=1}^N \text{Received update}_i + \text{Correction}_i$. Yet, to be able to broadcast it back, it compresses it, *before* applying the update. We refer to this strategy as the *degraded update* approach. Its major advantage is simplicity, and it was used in all previous papers performing double compression. As this appears to be a waste of valuable information, we consider instead updating the global model w_{k+1} independently of the downlink compression:

$$w_{k+1} = w_k - \gamma \frac{1}{N}\sum_{i=1}^N \text{Update}_i(\hat{w}_k) + \text{Correction}_i\,. \qquad \text{(Preserved Update)}$$

for a sequence $\hat{w}_k$ (a) which is updated *using a compressed message*, such that (b) $\mathbb{E}[\hat{w}_k] = w_k$, (*conditional unbiasedness*) and (c) $\text{Var}(\hat{w}_k|w_k) = \mathbb{E}[\|\hat{w}_k - w_k\|^2|w_k]$ is "small" (controlled *conditional variance*). Building such a sequence $\hat{w}_k$ is no easy task. For example, setting:

$$\hat{w}_k^{\spadesuit} := \mathcal{C}_{\text{dwn}}(w_k) \quad \text{or} \quad \hat{w}_k^{\clubsuit} := \hat{w}_{k-1}^{\clubsuit} + \mathcal{C}_{\text{dwn}}(w_k - \hat{w}_{k-1}^{\clubsuit})$$
$$\text{or} \quad \hat{w}_k^{\blacklozenge} := \hat{w}_{k-1}^{\blacklozenge} + \mathcal{C}_{\text{dwn}}\left(\gamma \frac{1}{N}\sum_{i=1}^N \text{Update}_i(\hat{w}_k^{\blacklozenge}) + \text{Correction}_i\right)$$

would all satisfy (a) and (b), but would result in a large variance. To fix this issue, we introduce another auxiliary model, denoted H_k, and coined *downlink memory term*, which is available on both workers and central server. We define:

$$\begin{cases} \hat{w}_k^{\blacktriangledown} &= H_{k-1} + \mathcal{C}_{\text{dwn}}(w_k - H_{k-1}) \\ H_k &= H_{k-1} + \alpha_{\text{dwn}}\mathcal{C}_{\text{dwn}}(w_k - H_{k-1}). \end{cases} \qquad \text{(MCM-downlink)}$$

For the uplink communication and update of the main sequence w_k, we use (Preserved Update), with Update_i and Correction_i as in Chapter 2 to tackle heterogeneity

$$\begin{cases} w_{k+1} &= w_k - \frac{\gamma}{N}\sum_{i=1}^N \mathcal{C}_{\text{up}}(g_{k+1}^i(\hat{w}_k^{\blacktriangledown}) - h_k^i) + h_k^i \\ h_{k+1}^i &= h_k^i + \alpha_{\text{up}}\mathcal{C}_{\text{up}}(g_{k+1}^i(\hat{w}_k^{\blacktriangledown}) - h_k^i) \qquad \forall i \in [\![1, N]\!] \end{cases} \qquad \text{(MCM-uplink)}$$

MCM consists in alternating (MCM-uplink) and (MCM-downlink), starting from $\hat{w}_0 = H_0 = w_0$. In the following, we simple denote $\hat{w}_k$ for $\hat{w}_k^{\blacktriangledown}$, when no confusion can be made.

Rate α_{dwn}. It is necessary to use $\alpha_{\text{dwn}} < 1$. Otherwise, the compression noise tends to propagate and is amplified, because of the multiplicative nature of the compression. In fact, for $\alpha_{\text{dwn}} = 1$, we get $H_k \equiv \hat{w}_k$ and $\hat{w}_k^{\blacktriangledown} \equiv \hat{w}_k^{\clubsuit}$ (and for $\alpha_{\text{dwn}} = 0$, we get $H_k \equiv 0$ and $\hat{w}_k^{\blacktriangledown} \equiv \hat{w}_k^{\spadesuit}$) and both these solutions are not efficient. In Figure 3.1 we compare on a numerical experiment the algorithms when using $\hat{w}^{\spadesuit}$, $\hat{w}^{\blacklozenge}$, $\hat{w}^{\clubsuit}$ and $\hat{w}^{\blacktriangledown}$ (MCM), showing that only the latest converges.

Related work on Perturbed iterate analysis Analysis relies on the theory of perturbed iterate, introduced by Mania et al. [77] to deal with asynchronous SGD. More recently, it was used by Stich and Karimireddy [113], Gorbunov et al. [39] to analyze the convergence of algorithms with uplink compressions, error feedback and asynchrony. When th support point $\hat{w}_k$ satisfies $\mathbb{E}[\hat{w}_k|w_k] = w_k$, i.e., we are using gradients at randomly perturbed points, this can also be seen as a form of randomized smoothing [105].

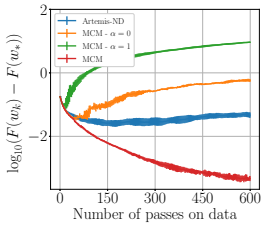
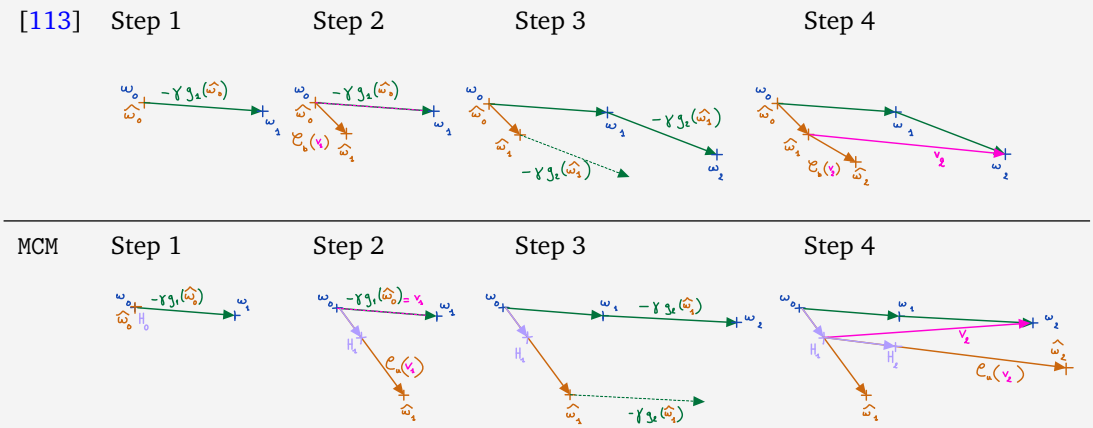


Figure 3.1: Comparing $\hat{w}^{\spadesuit,\clubsuit,\blacklozenge,\heartsuit}$

3.1.1 MCM vs Error Feedback

As mentioned in the introductory paragraph of this Chapter, our approach is related to *error feedback* techniques, as introduced by Seide et al. [107], and explored by [113]. Those techniques have been used mostly with contractive compression operators, that are generally biased (e.g., Top-k compressor, rescaled quantization, etc.) leading to some improvement in the context of double compression Zheng et al. [135], Tang et al. [118] under much stronger assumptions (uniformly bounded gradients and homogeneous tasks). For unbiased compression, as considered in Dore, it did not lead to any theoretical improvement [Remark 2 in Sec. 4.1., 74]. On the other hand, *we combine three sequences:* the preserved iterate sequence (w_k), the memories (H_k), and the sequence of support points $\hat{w}_k$. This makes the specificity of our approach. Relying on the auxiliary variable H_k that we introduce appears to be key to obtaining the desired properties for $\hat{w}_k$, especially *conditional unbiasedness* and *controlled variance*.

In the following plots, we instantiate the comparison between the approach in Stich and Karimireddy [113] and MCM. For representation and comparison purposes, we ignore the distribution ($N = 1$ on the plot) and the uplink compression on that representation. We use an unbiased compression $\mathcal{C}_u$ while [113] uses a biased contractive compression operator, $\mathcal{C}_b$, that can be obtained as $\frac{1}{\omega+1}\mathcal{C}_u$.



The main steps are recalled hereafter:
In [113], **Step 1:** we start from $w_0 = \hat{w}_0$. We apply the update $-\gamma g_1(w_0)$ to obtain w_1. **Step 2:** we compute $v_1 = w_1 - \hat{w}_0$, and compress it with $\mathcal{C}_b$, and update $\hat{w}_1 = \hat{w}_0 + \mathcal{C}_b(v_1)$. **Step 3:** we compute $-\gamma g_2(\hat{w}_1)$, and apply it to w_1, to get $w_2 = w_1 - \gamma g_2(\hat{w}_1)$.
Step 4: we compute $v_2 = w_2 - \hat{w}_1$, we compress v_2 with $\mathcal{C}_b$, and update $\hat{w}_2 = \hat{w}_1 + \mathcal{C}_b(v_2)$.

In MCM, **Step 1:** *same step*, and we let $H_0 = w_0$.
Step 2: we compute $v_1 = w_1 - H_0$, and compress it with $\mathcal{C}_u$, and update $\hat{w}_1 = H_0 + \mathcal{C}_u(v_1)$, we have $\mathbb{E}[\hat{w}_1] = w_1$. We update $H_1 = H_0 + \alpha_{\mathrm{dwn}}\mathcal{C}_u(v_1)$, $\alpha_{\mathrm{dwn}} < 1$.
Step 3: we compute $-\gamma g_2(\hat{w}_1)$, and apply it to w_1, to get $w_2 = w_1 - \gamma g_2(\hat{w}_1)$.
Step 4: we compute $v_2 = w_2 - H_1$, we compress v_2 with $\mathcal{C}_u$, and update $\hat{w}_2 = H_1 + \mathcal{C}_b(v_2)$, we have $\mathbb{E}[\hat{w}_2|w_2] = w_2$. We update $H_2 = H_1 + \alpha_{\mathrm{dwn}}\mathcal{C}_b(v_2)$.

Extension. In [C7], we also introduce a randomized variant of the algorithm Rand-MCM, in which each local worker is provided with a *different* local model at which the gradient is evaluated. We show to which extent this variant enables to improve the convergence. As the algorithm is originally only suitable in a *cross-silo* setting, as the central server has to maintain a copy of a model for each user, we propose ways to extend it to the *cross-device* setting.

3.2 Theoretical results

In this section, we provide a single representative result for the behavior of MCM, in the homogeneous case (i.e., with $F_i = F_j$ and $\alpha_{\mathrm{up}} = 0$), for strongly convex functions to highlight the impact of the memory mechanism on the downlink compression.

For k in $\mathbb{N}$, we define a potential $V_k = \mathbb{E}[\|w_k - w_*\|^2] + 32\gamma L\omega_{\mathrm{dwn}}^2\mathbb{E}[\|w_k - H_{k-1}\|^2]$, which serves as Lyapunov function. V_k is composed of two terms: the first one controls the quadratic distance to the optimal model, and the second controls the variance of the local models $\hat{w}_k$. We choose $\alpha_{\mathrm{dwn}} = (8\omega_{\mathrm{dwn}})^{-1}$, and denote

$$\Phi(\gamma) := (1 + \omega_{\mathrm{up}})\left(1 + 64\gamma L\omega_{\mathrm{dwn}}^2\right).$$

Theorem 3.1 (Convergence of MCM in the homogeneous and strongly-convex case)**.** *Under A1, A3, 4 and 6, with $\mu > 0$, for k in $\mathbb{N}$, for any sequence $(\gamma_k)_{k\geqslant 0} \leqslant \gamma_{\max}$ we have:*

$$V_k \leqslant (1 - \gamma_k\mu)V_{k-1} - \gamma_k\mathbb{E}\left[F(\widehat{w}_{k-1}) - F(w_*)\right] + \frac{\gamma_k^2\sigma^2\Phi(\gamma_k)}{Nb}, \tag{3.1}$$

Consequently,

1. *if $\sigma^2 = 0$ (noiseless case), for $\gamma_k \equiv \gamma_{\max}$ we recover a linear convergence rate: $\mathbb{E}[\|w_k - w_*\|^2] \leqslant (1 - \gamma_{\max}\mu)^k V_0$;*
2. *if $\sigma^2 > 0$, taking for all K in $\mathbb{N}$, $\gamma_K = 2/(\mu(K+1)+\widetilde{L})$, for the weighted Polyak-Ruppert average $\bar{w}_K = \sum_{k=1}^K \lambda_k w_{k-1} / \sum_{k=1}^K \lambda_k$, with $\lambda_k := (\gamma_{k-1})^{-1}$,*

$$\mathbb{E}\left[F(\bar{w}_K) - F(w_*)\right] \leqslant \frac{\mu + 2\widetilde{L}}{4\mu K^2}\|w_0 - w_*\|^2 + \frac{4\sigma^2(1+\omega_{\mathrm{up}})}{\mu K N b}\left(1 + \frac{64L\omega_{\mathrm{dwn}}^2}{\mu K}\ln(\mu K + \widetilde{L})\right). \tag{3.2}$$

Reduced Limit Variance (Equation (3.1)). For a constant γ, the variance term (i.e., term proportional to σ^2) in Equation (3.1) is upper bounded by $\frac{\gamma^2\sigma^2}{Nb}(1+\omega_{\mathrm{up}})(1+64\gamma L\omega_{\mathrm{dwn}}^2)$. The impact of the downlink compression is attenuated by a factor γL. As γ decreases, this makes the limit variance similar to the one of Diana, i.e., without downlink compression [81, Eq. 16 in Th. 2] and much lower than the variance for previous algorithms using double compression for which the variance scales quadratically with the compression constants as

$\gamma^2\sigma^2(1+\omega_{\rm up})(1+\omega_{\rm dwn})/N$: (1) for `Dore`, see Corollary 1 in Liu et al. [74] (who indicate $(1-\rho)^{-1} \geqslant (1+\omega_{\rm up}/N)(1+\omega_{\rm dwn})$), (2) for `Artemis` see Theorem 2.2, (3) for Gorbunov et al. [39], see Theorem I.1. (with $\gamma D_1' \propto \gamma^2\sigma^2(1+\omega_{\rm up})(1+\omega_{\rm dwn})/N$).

Bound 3.2 has a quadratic dependence on $\omega_{\rm dwn}$, but the corresponding term is divided by an extra factor K, the number of iterations.

Convergence and complexity: With a decaying sequence of steps, we obtain a convergence rate scaling as $O(K^{-1})$ in Equation (3.2), without dependency on the $\omega_{\rm dwn}$ in the dominating term, which only appears in faster decaying terms scaling as K^{-2}. The iteration complexity (i.e., number of iterations to achieve ε expected error) is thus at first order $O_{\varepsilon\to 0}(\frac{\sigma^2(1+\omega_{\rm up})}{\mu\varepsilon Nb})$. Again, this matches the complexity of Diana [53, see Theorem 1 and Corollary 1] and is smaller by a factor $1+\omega_{\rm dwn}$ than the one of `Artemis`, `Dore`, `DIANAsr-DQ` (see Corollary I.1. in [39]).

Extensions. In [C7], we also include convergence results first, in the heterogeneous case; and second, under weaker assumptions on the loss function, e.g., in the non-strongly convex and non convex cases. We also discuss the impact on the maximal learning rate. We summarize those elements in the following table:

Problem		Diana	Artemis, Dore	MCM, Rand-MCM
	$L\gamma_{\max} \propto$	$1/(1+\omega_{\rm up})$	$1/(1+\omega_{\rm up})(1+\omega_{\rm dwn})$	$1/(1+\omega_{\rm dwn})\sqrt{1+\omega_{\rm up}} \wedge 1/(1+\omega_{\rm up})$
	Lim. var. $\propto \gamma^2\sigma^2/n\times$	$(1+\omega_{\rm up})$	$(1+\omega_{\rm up})(1+\omega_{\rm dwn})$	$(1+\omega_{\rm up})(1+\gamma L\omega_{\rm dwn}^2)$
μ-strg-convex	Rate on init. cond.	$(1-\gamma\mu)^k$	$(1-\gamma\mu)^k$	$(1-\gamma\mu)^k$
	Complexity	$(1+\omega_{\rm up})/\mu\varepsilon N$	$(1+\omega_{\rm dwn})(1+\omega_{\rm up})/\mu\varepsilon N$	$(1+\omega_{\rm up})/\mu\varepsilon N$
Convex	Complexity	$(\omega_{\rm up}+1)/\varepsilon^2$	$(1+\omega_{\rm up})(1+\omega_{\rm dwn})/\varepsilon^2$	$(\omega_{\rm up}+1)/\varepsilon^2$

3.3 Experiments

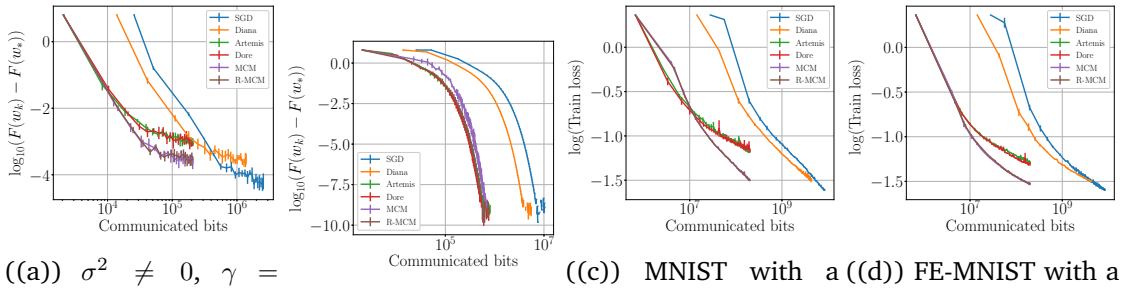
In this section, we illustrate the validity of the theoretical results given in the previous section on both synthetic and real datasets, on (1) least-squares linear regression (LSR), (2) logistic regression (LR), and (3) non-convex deep learning. We compare `MCM` with classical algorithms used in distributed settings: `Diana`, `Artemis`, `Dore` and of course the simplest setting - `SGD`, which is the baseline.

In these experiments, we provide results on the log of the excess loss $F(w_k) - F_*$, averaged on 5 runs (resp. 2) in convex settings (resp. deep learning), with errors bars displayed on each figure, corresponding to the standard deviation of $\log_{10}(F(w_k) - F_*)$. On Figure 3.3, the X-axis is respectively the number of iterations and the number of bits exchanged.

Each experiment has been run with $N = 20$ workers using stochastic scalar quantization [4], w.r.t. 2-norm. To maximize compression, we always quantize on a single level ($s = 2^0$), unless for PP ($s = 2^1$) and neural network (the value of s depends on the dataset). We used 10 datasets.

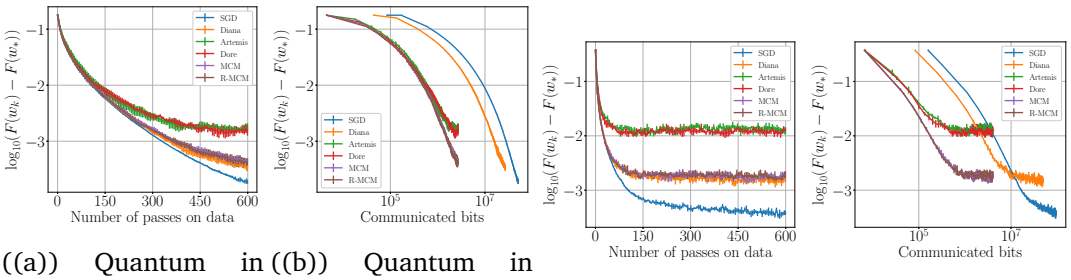
- One toy dataset devoted to linear regression in an homogeneous setting. This toy dataset allows to illustrate `MCM` properties in a simple framework, and in particular to ilustrate that when $\sigma^2 = 0$, we recover a linear convergence[1], see Figure 3.2(b).

[1] Even stronger, we show in experiments that we recover a linear rate if we have $\sigma_* = 0$ (the noise over stochastic gradient computation at the optimum point $w_\star$).



((a)) $\sigma^2 \neq 0$, $\gamma = (L\sqrt{k})^{-1}$ ((b)) $\sigma^2 = 0$, $\gamma = L^{-1}$ ((c)) MNIST with a CNN ((d)) FE-MNIST with a CNN

Figure 3.2: Convergence on neural networks.



((a)) Quantum in #iter. ((b)) Quantum in #bits ((c)) A9A in #iter. ((d)) A9A in #bits

Figure 3.3: Experiments on real dataset with $\gamma = 1/L$, quantization with $s = 1$, LSR (a,b), LR (c,d).

- Five datasets commonly used in convex optimization (a9a, quantum, phishing, superconduct and w8a). Experiments were conducted for heterogeneous tasks obtained by clustering inputs.
- Four dataset in a non-convex settings (CIFAR10, Fashion-MNIST, FE-MNIST, MNIST).

All experiments are performed without any tuning of the algorithms, (e.g., with the same learning rate for all algorithms and without reducing it after a certain number of epochs). Indeed, our goal is to show that our method achieves a performance close to the unidirectional-compression framework (`Diana`), while performing an important downlink compression.

On Figure 3.3, we display the excess loss for *quantum* and *a9a* w.r.t. the number of iteration and number of communicated bits. Similar plots of *phising*, *superconduct* and *w8a* can be found on our github repository. We report their excess loss after 450 iterations in Table 3.1.

Table 3.1: `MCM`- convex experiments, b is the batch size

Excess loss after 450 epochs	`SGD`	`Diana`	`MCM`	`Dore`	Ref
a9a ($b = 50$)	-3.5	-2.7	-2.7	-1.8	[13]
quantum ($b = 400$)	-3.4	-3.2	-3.2	-2.6	[12]
phishing ($b = 50$)	-3.7	-3.5	-3.4	-2.7	[13]
superconduct ($b = 50$)	-1.6	-1.6	-1.55	-1.45	[49]
w8a ($b = 12$)	-3.5	-3.0	-2.5	-1.75	[13]
Compression	no	uni-dir	bi-dir	bi-dir	

Saturation level. All experiments are performed with a *constant learning rate* γ to observe the bias (initial reduction) and the variance (saturation level) independently. Stochastic gradient descent results in a fast convergence during the first iterations, and then reaches a saturation at a given level proportional to σ^2. Theory states that the

variance of MCM is proportional to ω_{up}, this is experimentally observed on Tables 3.1 and 3.2 and figures 3.2 and 3.3: MCM meets Diana while Artemis and Dore saturate at a higher level (scaling as $\omega_{\text{up}} \times \omega_{\text{dwn}}$). These trade-offs are preserved with optimized learning rates.

Linear convergence when $\sigma^2 = 0$. The six algorithms present a linear convergence when $\sigma^2 = 0$. This is illustrated by Figure 3.2(b): we ran experiments with a full gradient descent. Note that in these settings MCM has a slightly worse performance than other methods; however, this slow-down is compensated by Rand-MCM.

Deep learning. Table 3.2 and figures 3.2(c) and 3.2(d) illustrate experiments with neural networks. Again, MCM meets Diana rates as anticipated by the theory in the non-convex case.

Table 3.2: Accuracy and train loss in non-convex experiments.

	Algorithm	MNIST	Fashion MNIST	FE-MNIST	CIFAR-10
Accuracy after 300 epochs	SGD:	99.0%	92.4%	99.0%	69.1%
	Diana:	98.9%	92.4%	98.9%	64.0%
	MCM:	98.8%	90.6%	98.9%	63.5%
	Artemis:	97.9%	86.7%	98.3%	54.8%
	Dore:	97.9%	87.9%	98.5%	56.3%
Train loss after 300 epochs	SGD:	0.025	0.093	0.026	0.909
	Diana:	0.034	0.141	0.031	1.047
	MCM:	0.033	0.209	0.030	1.096
	Artemis:	0.075	0.332	0.052	1.342
	Dore:	0.072	0.300	0.048	1.292

Overall, these experiments show the benefits of MCM and Rand-MCM, that reach the saturation level of Diana while exchanging at 10x to 100x fewer bits.

Again, we refer to the paper for complementary experiments (e.g. with partial participation for Rand-MCM), experimental details, etc. All the code is provided on our github repository.

Conclusion In this chapter, we described a new algorithm to perform bidirectional compression while achieving the convergence rate of algorithms using compression in a single direction. With MCM we stress the importance of not degrading the global model, a different point of view on error feedback schemes. In addition, we add the concept of randomization which allows to reduce the variance associated with the downlink compression. The analysis of MCM is more challenging as the algorithm involves perturbed iterates. Proposing such an analysis is also key to unlocking other challenges in distributed learning, e.g., proposing practical algorithms for partial participation, incorporating privacy-preserving schemes *after* the global update is performed, dealing with local steps, etc. This approach could also be pivotal in non-smooth frameworks, as it can be considered as a weak form of randomized smoothing.

Part II

First order optimization

4

PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python

This chapter corresponds to the preprint [40], currently under review. My PhD student Baptiste Goujaud is the main contributor of the package, which is a joint work with Céline Moucer, François Glineur, Julien M. Hendrickx, Adrien B. Taylor, **AD**.

This package was crucial to our recent work [P4], mentioned at the end of the chapter.

Contribution: PEPit is a package enabling computer-assisted worst-case analyses of first-order optimization methods. We cast the problem of performing a worst-case analysis, often referred to as a performance estimation problem (PEP), as a semidefinite program (SDP) which can be solved numerically. To obtain such a guarantee, the package users only have to write FOM nearly as they would have implemented them. The package then takes care of the SDP modelling parts, and the worst-case analysis is performed numerically.
The package enables users to easily obtain worst-case analyses for most of the standard FOM, class of problems, performance measures and initial conditions. This is useful to numerically verify existing convergence guarantees, as well as to ease the development of new analyses and methods. To this end, the toolbox contains tools for analyzing classical scenarios of the first-order literature: standard problem classes (such as convex functions, smooth convex functions, Lipschitz convex functions, etc.) and algorithmic operations (such as gradient, proximal, or linear optimization oracles, etc.).
The package contains more than 50 examples and is designed in an open fashion, allowing users to easily add new ingredients (e.g., their own problem classes, oracles, or algorithms).

Introduction. Due to their low cost per iteration, first-order optimization methods (FOM) became a major tool in the modern numerical optimization toolkit. Those methods are particularly well suited when targeting only low to medium accuracy solutions, and play a central role in many fields of applications that include machine learning and signal processing. Their simplicity further allows both occasional and expert users to use them. On the contrary, when it comes to their analyses (usually based on worst-case scenarios), they are mostly reserved to expert users. Our work allows a simpler and reproducible

access to worst-case analyses for FOM.

PEPit is a `python` package enabling computer-assisted worst-case analysis of a large family of FOM. After being provided with a first-order method and a standard problem class, the package reformulates the problem of performing a worst-case analysis as a semidefinite program (SDP). This technique is commonly referred to as *performance estimation problems* (PEPs) and was introduced by [28, 27]. The package uses PEPs as formalized by [123, 121].

In short, performing a worst-case analysis of a first-order algorithm usually relies on four main ingredients: a FOM (to be analyzed), a class of problems (containing the assumptions on the function to be minimized), a performance measure (measuring the quality of the output of the algorithm under consideration; for convenience here we assume that the algorithm aims at minimizing this performance measure and our analysis aims at finding a worst-case guarantee on it), and an initial condition (measuring the quality of the initial iterate). Performing the worst-case analysis (i.e., computing worst-case scenarios) corresponds to maximizing the performance measure of the algorithm on the class of problems, under a constraint induced by the initial condition. It turns out that such optimization problems can often be solved using SDPs in the context of FOM.

PEPs provide a principled approach to worst-case analyses, but usually relies on potentially tedious semidefinite programming (SDP) modelling and coding steps. The PEPit package eases the access to the methodology by automatically handling the modelling part, thereby limiting the amount of time spent on this tedious task and the risk of introducing coding mistakes in the process. In short, this work allows users to (i) write their first-order algorithms nearly as they would have implemented them, and (ii) let PEPit (a) perform the modelling and coding steps, and (b) perform the worst-case analysis numerically using tools for semidefinite programming in `python` [82, 20, 89].

Related works. The PEPit package relies on performance estimation problems as formalized in [121]. It also contains some improvements and generalization to other problem and algorithmic classes such as monotone and nonexpansive operators [102, 73], stochastic methods and verification of potential (or Lyapunov/energy) functions [54, 30, 119] as inspired by the related control-theoretic IQC framework [69]. The package also contains numerous examples; e.g., recent analyses and developments from [62, 125, 48, 63, 61, 73, 34, 120, 1]. The package can be seen as an extended open source `python` version of the `matlab` package `PESTO` [122].

This chapter introducing PEPit is organized as follows. We first exemplify the PEP approach on a very simple example, namely computing a *worst-case contraction factor for gradient descent*, and show how to code this example in PEPit, we then give a broader introduction to the contents of the package and mention other numerical examples.

4.1 PEPit: illustration on a simple example

In this section, we illustrate the use of the package for studying the worst-case properties of a standard scenario: gradient descent for minimizing a smooth strongly convex function. The goal of this elementary example is twofold. First, we want to provide the base mathematical steps enabling the use of semidefinite programming for performing worst-case analyses, together with a corresponding PEPit code. Second, we want to highlight the main ingredients that can be generalized to other problem setups (e.g., Theorem 4.1 below

providing "interpolation conditions" for the class of smooth strongly convex functions), allowing to analyze more algorithms under different assumptions (which are listed in Section 4.2).

For this example, we consider the convex optimization problem

$$\min_{x\in\mathbb{R}^d} f(x), \tag{4.1}$$

where f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, or $f \in \mathcal{F}_{\mu,L}$ when d is unspecified). So we assume f to satisfy

1. (L-smoothness) $\forall x, y \in \mathbb{R}^d$ we have that $f(x) \leqslant f(y) + \langle \nabla f(y); x - y\rangle + \frac{L}{2}\|x - y\|_2^2$,
2. (μ-strong convexity) $\forall x, y \in \mathbb{R}^d$ we have that $f(x) \geqslant f(y)+\langle \nabla f(y); x-y\rangle+\frac{\mu}{2}\|x-y\|_2^2$.

Our goal for the rest of this section is to show how to compute the smallest possible $\tau(\mu, L, \gamma)$ (often referred to as the "contraction factor") such that

$$\|x_1 - y_1\|_2^2 \leqslant \tau(\mu, L, \gamma)\|x_0 - y_0\|_2^2, \tag{4.2}$$

is valid for all $f \in \mathcal{F}_{\mu,L}$ and all $x_0, y_0 \in \mathbb{R}^d$ when x_1 and y_1 are obtained from gradient steps from respectively x_0 and y_0. That is, $x_1 = x_0 - \gamma\nabla f(x_0)$ and $y_1 = y_0 - \gamma\nabla f(y_0)$. First, we show that the problem of computing $\tau(\mu, L, \gamma)$ can be framed as a semidefinite program (SDP), and then illustrate how to use PEPit for computing it without going into the SDP modelling details.

4.1.1 A performance estimation problem for the gradient method

It is relatively straightforward to establish that the smallest possible $\tau(\mu, L, \gamma)$ for which (4.2) is valid can be computed as the worst-case value of $\|x_1 - y_1\|_2^2$ when $\|x_0 - y_0\|_2^2 \leqslant 1$. That is, we compute $\tau(\mu, L, \gamma)$ as the optimal value to the following optimization problem:

$$\begin{aligned}\tau(\mu, L, \gamma) = \max_{\substack{f,d\\ x_0,x_1,y_0,y_1\in\mathbb{R}^d}} \quad & \|x_1 - y_1\|_2^2 \\ \text{s.t. } & d \in \mathbb{N}, \quad \|x_0 - y_0\|_2^2 \leqslant 1, \\ & f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d), \quad x_1 = x_0 - \gamma\nabla f(x_0), \quad y_1 = y_0 - \gamma\nabla f(y_0).\end{aligned} \tag{4.3}$$

As written in (4.3), this problem involves an infinite-dimensional variable f. Our first step towards formulating (4.3) as an SDP consists in reformulating it by sampling f (i.e., evaluating its function value and gradient) at the two points where its gradient is evaluated. The condition $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d),\ x_1 = x_0 - \gamma\nabla f(x_0),\ y_1 = y_0 - \gamma\nabla f(y_0)$ in (4.3) is replaced by:

$$\exists f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d) : \left\{ \begin{array}{ll} f(x_0) = f_{x_0} & \nabla f(x_0) = g_{x_0} \\ f(y_0) = f_{y_0} & \nabla f(y_0) = g_{y_0} \end{array} \right. , \quad x_1 = x_0 - \gamma g_{x_0}, \quad y_1 = y_0 - \gamma g_{y_0}, \tag{4.4}$$

where we replaced the variable f by its discrete version, which we constrain to be "interpolable" (or "extendable") by a smooth strongly convex function over $\mathbb{R}^d$. The $\max$ is then taken over $d, f_{x_0}, f_{y_0},\ (x_0, x_1, g_{x_0}, y_0, y_1, g_{y_0}) \in \mathbb{R}^d$. To arrive at a tractable problem, we use the following interpolation (or extension) result.

Theorem 4.1. *[123, Theorem 4] Let I be an index set and $S = \{(x_i, g_i, f_i)\}_{i\in I}$ be such that $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. There exists a function $F \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ such that $f_i = F(x_i)$ and $g_i = \nabla F(x_i)$ (for all $i \in I$) if and only if for all $i, j \in I$ we have*

$$f_i \geqslant f_j + \langle g_j; x_i - x_j\rangle + \tfrac{1}{2L}\|g_j - g_i\|_2^2 + \tfrac{\mu L}{2(L-\mu)}\|x_i - x_j - \tfrac{1}{L}(g_i - g_j)\|_2^2. \tag{4.5}$$

Using Theorem 4.1, we can formulate the problem of computing $\tau(\mu, L, \gamma)$ as a (non-convex) quadratic problem:

$$\begin{aligned}\max_{\substack{d, f_{x_0}, f_{y_0} \\ x_0, g_{x_0}, y_0, g_{y_0} \in \mathbb{R}^d}} \quad & \|(x_0 - \gamma g_{x_0}) - (y_0 - \gamma g_{y_0})\|_2^2 \\ \text{s.t. } & d \in \mathbb{N}, \quad \|x_0 - y_0\|_2^2 \leqslant 1, \\ & f_{y_0} \geqslant f_{x_0} + \langle g_{x_0}; y_0 - x_0\rangle + \tfrac{1}{2L}\|g_{y_0} - g_{x_0}\|_2^2 + \tfrac{\mu L}{2(L-\mu)}\|x_0 - y_0 - \tfrac{1}{L}(g_{x_0} - g_{y_0})\|_2^2, \\ & f_{x_0} \geqslant f_{y_0} + \langle g_{y_0}; x_0 - y_0\rangle + \tfrac{1}{2L}\|g_{y_0} - g_{x_0}\|_2^2 + \tfrac{\mu L}{2(L-\mu)}\|x_0 - y_0 - \tfrac{1}{L}(g_{x_0} - g_{y_0})\|_2^2.\end{aligned}$$

Relying on a standard trick from semidefinite programming, one can convexify this problem using a Gram representation of the variable (this is due to maximization over d). That is, we formulate the problem using a positive semidefinite matrix $G \succcurlyeq 0$ defined as

$$G \triangleq \begin{pmatrix} \|x_0 - y_0\|_2^2 & \langle x_0 - y_0; g_{x_0}\rangle & \langle x_0 - y_0; g_{y_0}\rangle \\ \langle x_0 - y_0; g_{x_0}\rangle & \|g_{x_0}\|_2^2 & \langle g_{x_0}; g_{y_0}\rangle \\ \langle x_0 - y_0; g_{y_0}\rangle & \langle g_{x_0}; g_{y_0}\rangle & \|g_{y_0}\|_2^2 \end{pmatrix} \succcurlyeq 0.$$

Using this change of variable, we arrive to

$$\begin{aligned}\max_{f_{x_0}, f_{y_0}, G} \quad & G_{1,1} - 2\gamma(G_{1,2} - G_{1,3}) + \gamma^2(G_{2,2} + G_{3,3} - 2G_{2,3}) \\ \text{s.t. } & G \succcurlyeq 0, \quad G_{1,1} \leqslant 1, \\ & f_{y_0} \geqslant f_{x_0} + \tfrac{1}{L-\mu}\left(\tfrac{\mu L}{2}G_{1,1} - LG_{1,2} + \mu G_{1,3} + \tfrac{1}{2}G_{2,2} - G_{2,3} + \tfrac{1}{2}G_{3,3}\right), \\ & f_{x_0} \geqslant f_{y_0} + \tfrac{1}{L-\mu}\left(\tfrac{\mu L}{2}G_{1,1} - \mu G_{1,2} + LG_{1,3} + \tfrac{1}{2}G_{2,2} - G_{2,3} + \tfrac{1}{2}G_{3,3}\right),\end{aligned}$$

which can be solved numerically using standard tools, see, e.g., [20, 82]. Using numerical and/or symbolical computations, one gets $\tau(\mu, L, \gamma) = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$, i.e.

$$\|x_1 - y_1\|_2^2 \leqslant \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}\|x_0 - y_0\|_2^2, \tag{4.6}$$

for all $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ and $x_0, y_0 \in \mathbb{R}^d$ when $x_1, y_1 \in \mathbb{R}^d$ are generated from gradient steps from respectively x_0 and y_0. In the next section, we show how to perform this analysis using PEPit, which automates the sampling and SDP-modelling procedures. In more complex settings where more functions need to be sampled and/or more iterates have to be taken into account, avoiding those steps allows to largely limit the probability of making a mistake in the process of performing the worst-case analysis (numerically), while sparing a significant amount of time.

4.1.2 Code

In the previous section, we introduced the PEP and SDP modelling steps for computing a tight worst-case contraction factor for gradient descent in the form (4.2). Although this particular SDP (4.1.1) might be solved analytically, many optimization methods lead to larger SDPs with more complicated structures. In general, we can reasonably only hope to solve them numerically. Next, we describe how to use PEPit for computing a *contraction factor* without explicitly going into the modelling steps. Compared to previous section, we allow ourselves to perform $n \in \mathbb{N}$ iterations and compute the smallest possible value of $\tau(\mu, L, \gamma, n)$ such that

$$\|x_n - y_n\|_2^2 \leqslant \tau(\mu, L, \gamma, n)\|x_0 - y_0\|_2^2, \tag{4.7}$$

where x_n and y_n are computed from n iterations of gradient descent with step-size γ starting from respectively x_0 and y_0. As illustrated in the previous section for the case $n = 1$, computing the smallest possible such $\tau(\mu, L, \gamma, n)$ is equivalent to computing the worst-case value of $\|x_n - y_n\|_2^2$ under the constraint that $\|x_0 - y_0\|_2^2 \leqslant 1$ (note that we naturally have that $\tau(\mu, L, \gamma, n) \leqslant (\tau(\mu, L, \gamma, 1))^n$). This is what we do in the following lines using PEPit.

```python
from PEPit import PEP
from PEPit.functions import\
SmoothStronglyConvexFunction

problem = PEP()
```

Imports. Before going into the example, we have to include the right `python` imports.

```python
L = 1.           # Smoothness
mu = .1          # Strong convexity
gamma = 1. / L   # Step size
n = 1            # Nb. of iterations
```

Initialization of PEPit. First, we initialize a PEP object. This object enables manipulating the forthcoming ingredients of the PEP, such as functions and iterates.

For the sake of the example, let us pick some simple values for the problem class and algorithmic parameters, for which we perform the worst-case analysis.

```python
# Declare an L-smooth mu-strongly
# convex function named "func"
func = problem.declare_function(
SmoothStronglyConvexFunction,
mu=mu, # Strong convexity param.
L=L)   # Smoothness param.
```

Specifying the problem class. Second, we specify our working assumptions on the function to be optimized, and instantiate a corresponding object. Here, the minimization problem at hand was of the form (4.1) with a smooth strongly convex function.

```python
# Declare two starting points
x_0 = problem.set_initial_point()
y_0 = problem.set_initial_point()

# Initial cond. ||x_0 - y_0||^2 <= 1
problem.set_initial_condition(
            (x_0 - y_0) ** 2 <= 1)
```

Algorithm initialization. Third, we can instantiate the starting points for the two gradient methods that we will run, and specify an *initial condition* on those points. To this end, two starting points x_0 and y_0 are introduced, one for each trajectory, and a bound on the initial distance between those points is specified as $\|x_0 - y_0\|^2 \leqslant 1$.

```python
# Initialize the algorithm
x = x_0
y = y_0
# Run n steps of GD for the 2 sequences
for _ in range(n):
# Replace x and y by their next value
x = x - gamma * func.gradient(x)
y = y - gamma * func.gradient(y)
  # calls to f'(x), f'(y)
```

Algorithm implementation. In this fourth step, we specify the algorithm in a natural format. In this example, we simply use the iterates (which are PEPit objects) as if we had to implement gradient descent in practice using a simple loop.

```python
# Set the performance metric to the
#  distance ||x_n - y_n||^2
problem.set_performance_metric((x-y)**2)
```

Setting up a performance measure. We specify the *metric* for which we compute a worst-case performance. Here, we chose to compute the worst-case value of $\|x_n - y_n\|^2$.

```python
# Solve the PEP
pepit_tau = problem.solve()
```

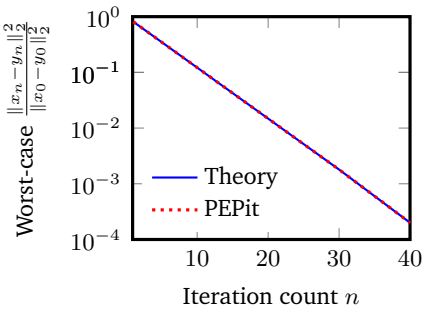
Solving the PEP. This asks PEPit to perform the modelling steps and call an appropriate SDP solver to perform the worst-case analysis.

```
(PEPit) Setting up the pb: size of the main PSD matrix: 4x4
(PEPit) Setting up the pb: performance measure is minimum
        of 1 element(s)
(PEPit) Setting up the pb: Adding initial conditions
and general constraints ...
(PEPit) Setting up the pb: initial conditions and general
constraints (1 constraint(s) added)
(PEPit) Setting up the pb: interpolation conditions for
         1 function(s)
function 1 : Adding 2 scalar constraint(s) ...
function 1 : 2 scalar constraint(s) added
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (solver: SCS);
        optimal value: 0.810001
```

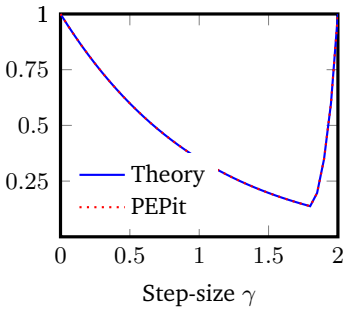
Output. Running the above code, we obtain (see PEPit/examples/) the following output. ($n = 1$, $L = 1$, $\mu = .1$ and $\gamma = 1$)

Note that the size of the SDP is larger than that of Section 4.1 (4×4 instead of 3×3 in (4.1.1)) because the modelling step is done in a slightly more generic way, which might not be exploiting all specificities of the problem at hand. For more complete examples of worst-case analyses using PEPit, see the examples.

It is also possible to run the code for different values of the parameters, as exemplified on Figure 4.1. This simple example allows to observe that numerical values obtained from PEPit match the worst-case guarantee (4.1(a)), and to optimize the step-size numerically in Figure 4.1(b).



((a)) Worst-case guarantee and theoretical tight bound as a function of the iteration count n for $\gamma = \frac{1}{L}$.



((b)) Worst-case guarantee and theoretical tight bound as a function of the step-size γ, at iteration $n = 5$.

Figure 4.1: Comparison: worst-case guarantee from PEPit and theoretical tight worst-case bound (4.6) for GD, in terms of $\frac{\|x_n-y_n\|_2^2}{\|x_0-y_0\|_2^2}$. Parameters are fixed to $\mu = 0.1$ and $L = 1$.

4.2 PEPit: general overview and content

Remark 4.2 (Important ingredients for the SDP reformulations)**.** *To understand what PEPit can do, it is crucial to understand what elements allowed to cast the worst-case analysis as such a semidefinite program (which is what we refer to as the "modelling" of the problem). In short, the SDP reformulation of the worst-case computation problem was made possible due to* 4 *main ingredients (see, e.g., [121, Section 2.2]):*

1. *the algorithmic steps can be expressed linearly in terms of the iterates and gradient values (i.e., step-sizes do not depend on the function at hand),*
2. *the class of functions has "interpolation condition"* [1] *that are linear in G and F,*
3. *the performance measure is linear (or convex piecewise linear) in G and F,*
4. *the initial condition is linear in G and F.*

Those ingredients allow using PEPs much beyond the simple setup of Section 4.1. That is, PEPs apply for performing worst-case analyses involving a variety of first-order oracles, initial conditions, performance measures, and problem classes.

In this section, describe the various choices of (i) elementary oracles used in algorithms, (ii) problem or function classes, (iii) performance measures, and (iv) initial conditions, that are naturally handled by PEPit. PEPit also allows studying methods for monotone inclusions and fixed point problems, but we do not cover them in this summary. In the optimization setting, the minimization problem under consideration has the form

$$F_\star \triangleq \min_{x\in\mathbb{R}^d} \left\{ F(x) \equiv \sum_{i=1}^{K} f_i(x) \right\}, \tag{4.8}$$

[1] Interpolation conditions characterize the **existence** of a function (that has particular function values and gradients at given points) in the considered class by a list of constraints on those gradients, points, and function values. Such interpolation theorems (see, e.g., Theorem 4.1) have been obtained in the literature for various problem classes, see, e.g., [123, 121, 102].

for some $K \in \mathbb{N}$ and where each f_i is assumed to belong to some class of functions denoted by $\mathcal{F}_i$, encoding our working assumptions on f_i. We further assume the algorithms to gather information about the functions $\{f_i\}_i$ only via black-box oracles such as gradient or proximal evaluations.

Black-box oracles. The base black-box optimization oracles/operations available in PEPit are the following:

- (sub)gradient steps,
- proximal steps,
- linear optimization (or conditional) steps

PEPit also allows for their slightly more general approximate/inexact and Bregman (or mirror) versions. Those oracles might be combined with a few other operations, such as exact line-search procedures.

Problem classes. A few base classes of functions are readily available:

- convex functions within different classes of assumptions possibly involving bounded gradients (Lipschitz functions), bounded domains, smoothness, and/or strong convexity. Those assumptions might be combined when compatible.
- Convex indicator functions, possibly with a bounded domain.
- Smooth nonconvex functions.

Beyond the pure optimization setting, PEPit also allows using operators within different classes of assumptions (namely: nonexpansive, Lipschitz, cocoercive, maximally monotone, and strongly monotone operators) for studying FOM for monotone inclusions and variational inequalities.

Performance measures and initial conditions. The package allows a large panel of performance measures and initial conditions. Essentially, everything that can be expressed linearly (or slightly beyond) in function values and quadratically in gradient/iterates (i.e., linear in the Gram representation of Section 4.1) might be considered. Following the notation of Section 4.1.2 (and denoting by $x_\star$ an optimal point of F), typical examples of initial conditions include:

- $\|x_0 - x_\star\|_2^2 \leqslant 1$,
- $\|\nabla F(x_0)\|_2^2 \leqslant 1$, †
- $F(x_0) - F(x_\star) \leqslant 1$,
- any linear combination of the above (see, e.g., examples in the *potential functions* folder).

Similarly, typical examples of performance measures include:

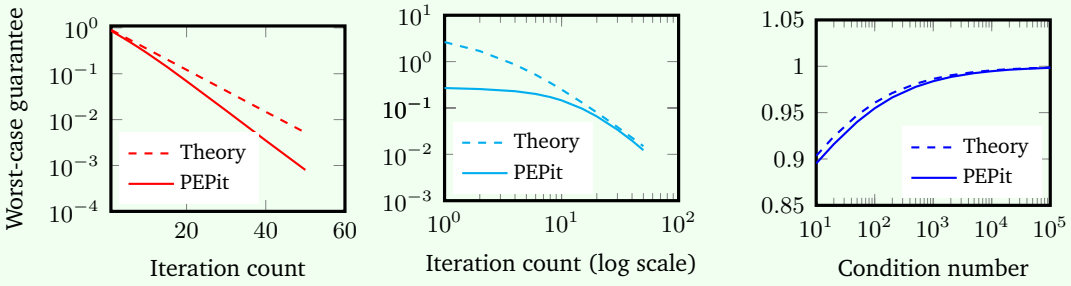
- $\|x_n - x_\star\|_2^2$,
- $\|\nabla F(x_n)\|_2^2$, †
- $F(x_n) - F(x_\star)$,
- linear combinations and minimum values of the above, e.g. $\min_{0 \leqslant i \leqslant n} \|\nabla F(x_i)\|_2^2$ †.

† (when F is differentiable, otherwise similar criterion involving some subgradient of F might be used).

Contributing. PEPit is designed for allowing users to easily contribute to add features to the package. Classes of functions (or operators) as well as black-box oracles can be implemented by following the contributing guidelines from the documentation. We also welcome any new example for analyzing a method/setting that is not already present in the toolbox.

Examples. PEPit contains about 50 examples that can readily be used, instantiating the different sets of black-box oracles, problem classes, and initial condition/performance measures. Those examples can be found in the folder PEPit/examples/

Extensions: In [P2], we propose detail some other examples, namely an accelerated gradient method [85], an accelerated Douglas-Rachford splitting [92], and point-SAGA [17] (a proximal method for finite-sum minimization). Graphical results are reported in Figure 4.2.



((a)) Accelerated gradient method: strong convexity parameter fixed to $\mu = 0.1$. Worst-case guarantee on $F(x_n) - F_\star$ as a function of n.

((b)) Accelerated Douglas-Rachford splitting with $\alpha = 0.9$ and $\mu = 0.1$. Worst-case guarantee on $F(x_n) - F_\star$ as a function of n.

((c)) Point-SAGA: $n = 5$ functions. Worst-case guarantee on $E_{j_t}[V(x_{j_t}^{(t+1)}]$ as a function of the condition number $\kappa = \frac{L}{\mu}$.

Figure 4.2: Comparisons between (numerical) worst-case bounds from PEPit and reference established theoretical worst-case guarantees. Dashed lines correspond to established worst-case bounds, where plain lines correspond to numerical worst-case guarantees from PEPit.

Conclusion. The PEPit package provides a simplified access to worst-case analyses of FOM in `python`. It implements the performance estimation approach while allowing to avoid the possibly heavy semidefinite programming modeling steps. The first version of the package already contains about 50 examples of FOM that can be analyzed through this framework. Those examples allow either reproducing or tightening, numerically, known worst-case analyses, or to provide new ones depending on the particular method and problem class at hand. Overall, we believe that this package allows to quickly (in)validate proofs (step towards reproducible theory) which should help both the development and the review process in optimization. It is also a nice pedagogical tool for learning algorithms together with their worst-case properties just by playing with them. Possible extensions under consideration include an option for searching for Lyapunov (or potential/energy) functions, as well as a numerical proof assistant, and to incorporate recent extensions of PEP/IQCs to distributed and decentralized optimization [116, 14].

Optimal convergence rates for a class of non smooth functions [P4], (B. Goujaud, A. Taylor, AD) In this recent work, we analyze worst-case convergence guarantees of FOM over a function class extending that of smooth and convex functions. This class contains convex functions that admit a simple quadratic upper bound. Its study is motivated by its stability under minor perturbations. We provide a thorough analysis of first-order methods, including worst-case convergence guarantees for several methods,

and demonstrate that some of them achieve the optimal worst-case guarantee over the class. We support our analysis by numerical validation of worst-case guarantees using performance estimation problems. A few observations can be drawn from this analysis, particularly regarding the optimality (resp. and adaptivity) of the heavy-ball method (resp. heavy-ball with line-search). Finally, we show how our analysis can be leveraged to obtain convergence guarantees over more complex classes of functions. Overall, this study brings insights on the choice of function classes over which standard first-order methods have working worst-case guarantees.

5

Quadratic minimization: a testbed for first order optimization

This chapter is composed by two main parts, corresponding to two recent works: [W1], *Quadratic minimization: From conjugate gradient to an adaptive Heavy-ball method with Polyak step-sizes* Baptiste Goujaud, Adrien Taylor, **A.D.**, (under review, also accepted to Neurips optimization workshop 2022), and [C9], *Super-acceleration with cyclical step-sizes*, B. Goujaud, D. Scieur, **A.D.**, A. Taylor, and F. Pedregosa, Aistats 2022.

In both these papers, we focus on the problem of optimizing quadratic functions.

5.1 From conjugate gradient to an adaptive Heavy-ball method with Polyak step-sizes

In this work, we propose an adaptive variation on the classical Heavy-ball method for convex quadratic minimization. The adaptivity crucially relies on so-called "Polyak step-sizes", which consists in using the knowledge of the optimal value of the optimization problem at hand instead of problem parameters such as a few eigenvalues of the Hessian of the problem. This method happens to also be equivalent to a variation of the classical conjugate gradient method, and thereby inherits many of its attractive features, including its finite-time convergence, instance optimality, and its worst-case convergence rates. The classical gradient method with Polyak step-sizes is known to behave very well in situations in which it can be used, and the question of whether incorporating momentum in this method is possible and can improve the method itself appeared to be open. We provide a definitive answer to this question for minimizing convex quadratic functions, a arguably necessary first step for developing such methods in more general setups.

Consider the convex quadratic minimization problem in the form

$$\min_{x\in\mathbb{R}^d} \left\{ f(x) \triangleq \frac{1}{2}\langle x,\, Hx\rangle + \langle h,\, x\rangle \triangleq \frac{1}{2}\langle x - x_\star,\, H(x-x_\star)\rangle + f_\star \right\} \tag{5.1}$$

where $H \succcurlyeq 0$ is a symmetric positive semi-definite matrix, and we denote $f_\star$ the minimum value of f (a few instances of such problems are presented in Section 5.2.2). In the context of large-scale optimization (i.e. $d \gg 1$), we are often interested in using first-order iterative

methods for solving eq. (5.1). There are many known and celebrated iterative methods for solving such quadratic problems, including conjugate gradient, Heavy-ball methods (a.k.a., Polyak momentum), Chebyshev methods, and gradient descent. Each of those methods having different specifications, the choice of the method largely depends on the application at hand. In particular, a typical drawback of momentum-based methods is that they generally require the knowledge of some problem parameters (such as extreme values of the spectrum of H). This problem is typically not as critical for simpler gradient descent schemes with no momentum, although it generally still requires some knowledge on problem parameters if we want to avoid using linesearch-based strategies. This limitation motivates the search for adaptive strategies, fixing step-size using past observations about the problem at hand. In the context of (sub)gradient descent, a famous adaptive strategy is the so-called Polyak step-size, which relies on the knowledge of the optimal value $f_\star$:

$$x_{t+1} = x_t - \gamma_t \nabla f(x_t), \quad \text{with } \gamma_t = \frac{f(x_t) - f_\star}{\|\nabla f(x_t)\|^2}. \tag{5.2}$$

Polyak steps were originally proposed in Polyak [98] for nonsmooth convex minimization; it is also discussed in Boyd et al. [11] and a few variants are proposed by, e.g., [9, 75, 45] including for stochastic minimization. In terms of speed, this strategy (and variants) enjoy similar theoretical convergence properties as those for gradient descent. This methods appears to perform quite well in applications where $f_\star$ can be efficiently estimated—see, e.g., [50] for an adaptation of the method for estimating it online (although [50] contains a number of mistakes, the approach can be corrected to achieve the claimed target). Therefore, a remaining open question in this context is whether the performances of this method can be improved by incorporating momentum in it. A first answer to this question was provided by Barré et al. [9], although it is not clear that it can match the same convergence properties as optimal first-order methods.

In this work, we answer this question for the class of quadratic problems. In short, it turns out that the following conjugate gradient-like iterative procedure

$$x_{t+1} = \operatorname*{argmin}_x \left\{ \|x - x_\star\|^2 \text{ s.t. } x \in x_0 + \operatorname{span}\{\nabla f(x_0), \nabla f(x_1), \dots, \nabla f(x_t)\} \right\}, \tag{5.3}$$

can be rewritten exactly as an Heavy-ball type method whose parameters are chosen adaptively using the value of $f_\star$. This might come as a surprise, as the iteration eq. (5.3) might seem impractical due to its formulation relying on the knowledge of $x_\star$. More precisely, eq. (5.3) is exactly equivalent to:

$$x_{t+1} = x_t - (1 + m_t) \times h_t \nabla f(x_t) + m_t(x_t - x_{t-1}), \tag{5.4}$$

with parameters

$$\forall t \geqslant 0, \quad h_t \triangleq \frac{2(f(x_t) - f_\star)}{\|\nabla f(x_t)\|^2}, \tag{5.5}$$

$$m_0 \triangleq 0 \text{ and } \forall t \geqslant 1, \quad m_t \triangleq \frac{-(f(x_t) - f_\star)\langle \nabla f(x_t), \nabla f(x_{t-1})\rangle}{(f(x_{t-1}) - f_\star)\|\nabla f(x_t)\|^2 + (f(x_t) - f_\star)\langle \nabla f(x_t), \nabla f(x_{t-1})\rangle} \tag{5.6}$$

In eq. (5.4), m_t corresponds to the momentum coefficient and h_t to a step-size. With the tuning of eq. (5.5), this step-size is twice the Polyak step-size in eq. (5.2). This Heavy-ball momentum method with Polyak step-sizes is summarized in Algorithm 2 and illustrated in Figure 5.1. Due to its equivalence with eq. (5.3), the Heavy-ball method eq. (5.4) inherits nice advantageous properties of conjugate gradient-type methods, including:

(i) finite-time convergence: the problem eq. (5.1) is solved exactly after at most d iterations,

(ii) instance optimality: for *all* $H \succcurlyeq 0$, there is no first-order method satisfying $x_{t+1} \in x_0 + \text{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}$ that results in a smaller $\|x_t - x_\star\|$,

(iii) it inherits optimal worst-case convergence rates on quadratic functions.

Of course, a few of those points needs to be nuanced in practice due to finite precision arithmetic. The equivalence between eq. (5.3) and eq. (5.4) is formally stated in the following theorem.

Theorem 5.1. *Let $(x_t)_{t\in\mathbb{N}}$ be the sequence defined by the recursion eq. (5.3), namely such that for any t, x_{t+1} is the Euclidean projection of $x_\star$ onto the affine subspace $x_0 + \text{span}\{\nabla f(x_0), \nabla f(x_1), \dots, \nabla f(x_t)\}$. Then $(x_t)_{t\in\mathbb{N}}$ is the sequence generated by Algorithm 2.*

Algorithm 2: Adaptive Heavy-ball algorithm

Input T and $f : x \mapsto f(x) \triangleq \frac{1}{2}\langle x - x_\star, H(x - x_\star)\rangle + f_\star$
Initialize $x_0 \in \mathbb{R}^d$, $m_0 = 0$
for $t = 0 \cdots T - 1$ **do**
  $h_t = \frac{2(f(x_t) - f_\star)}{\|\nabla f(x_t)\|^2}$
  $x_{t+1} = x_t - (1 + m_t) \times h_t \nabla f(x_t) + m_t(x_t - x_{t-1})$
  $m_{t+1} = \frac{-(f(x_{t+1}) - f_\star)\langle \nabla f(x_{t+1}), \nabla f(x_t)\rangle}{(f(x_t) - f_\star)\|\nabla f(x_{t+1})\|^2 + (f(x_{t+1}) - f_\star)\langle \nabla f(x_{t+1}), \nabla f(x_t)\rangle}$;
end
Result: x_T

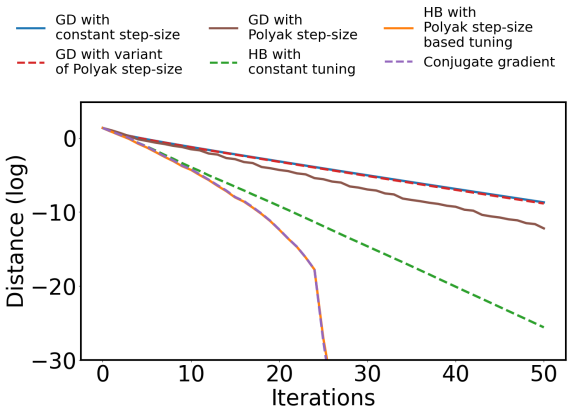


Figure 5.1: Comparison in semi-log scale over 50 iterations of different first-order methods applied on a 25-dimensional quadratic objective with condition number 10. **GD with constant step-size**, **GD with Polyak step-size** and **GD with variant of Polyak step-size** refer to the GD method tuned respectively with the step-size $\gamma = 2/(L + \mu)$, $\gamma_t = (f(x_t) - f_\star)/\|\nabla f(x_t)\|^2$ and $\gamma_t = 2(f(x_t) - f_\star)/\|\nabla f(x_t)\|^2$. **HB with constant tuning** is the HB method tuned with constant parameters $\gamma_t = (2/(\sqrt{L} + \sqrt{\mu}))^2$ and $m_t = ((\sqrt{L} - \sqrt{\mu})(\sqrt{L} + \sqrt{\mu}))^2$ while **HB with Polyak step-size based tuning** refers to Algorithm 2.

Theorem 5.1 turns out to be a particular case of a more general result stating that the iterates of any conjugate gradient-type method described with a polynomial Q as

$$x_{t+1} = \text{argmin}_x \left\{ \langle x - x_\star, Q(H)(x - x_\star)\rangle \text{ s.t. } x \in x_0 + \text{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\} \right\}, \tag{Q-minimization}$$

are equivalently written in terms of an adaptive Heavy-ball iteration. In particular, eq. (5.3) corresponds to eq. (Q-minimization) with $Q(x) = 1$. Similarly, classical conjugate gradient method corresponds to eq. (Q-minimization) with $Q(x) = x$ (this fact is quite famous, see, e.g., [87]). We were surprised not to find this general result written as is in the literature, and we therefore provide it in the main paper. The key point of this work is that the equivalent Heavy-ball reformulation of eq. (5.3) can be written in terms of $f_\star$, thereby obtaining a momentum-based Polyak step-size.

Proof and main theorem. [W1] contains the proof of the main result from which Theorem 5.1 is a corollary, and complementary experiments.

5.2 Super-Acceleration with Cyclical Step-size and clustered eigenvalues

Summary. In [C9], we develop a convergence-rate analysis of momentum with cyclical step-sizes. We show that under some assumption on the spectral gap of Hessians in machine learning, cyclical step-sizes are provably faster than constant step-sizes. More precisely, we develop a convergence rate analysis for quadratic objectives that provides optimal parameters and shows that cyclical learning rates can improve upon traditional lower complexity bounds. We further propose a systematic approach to design optimal first order methods for quadratic minimization with a given spectral structure. Finally, we provide a local convergence rate analysis beyond quadratic minimization for the proposed methods and illustrate our findings through benchmarks on least squares and logistic regression problems.

In this section, I will only describe the framework and the result for cycles of 2 steps.

Introduction. One of the most iconic methods in first order optimization is gradient descent with momentum, also known as the heavy ball method [97]. This method enjoys widespread popularity both in its original formulation and in a stochastic variant that replaces the gradient by a stochastic estimate, a method that is behind many of the recent breakthroughs in deep learning [117].

A variant of the stochastic heavy ball where the step-sizes are chosen in *cyclical* order has recently come to the forefront of machine learning research, showing state-of-the art results on different deep learning benchmarks [76, 110]. Inspired by this empirical success, we aim to study the convergence of the heavy ball algorithm where step-sizes $h_0, h_1, \ldots$ are not fixed or decreasing but instead chosen in cyclical order, as in Algorithm 3.

The heavy ball method with constant step-sizes enjoys a mature theory, where it is known for example to achieve optimal black-box worst-case complexity of quadratic convex optimization [84]. In stark contrast, little is known about the the convergence of the above variant with cyclical step-sizes. Our main motivating question is

Do cyclical step-sizes improve convergence of heavy ball?

Algorithm 3: Cyclical heavy ball $\mathrm{HB}_K(h_0, \dots, h_{K-1}; m)$

Input: Initialization x_0, momentum $m \in (0,1)$, step-sizes $\{h_0, \dots, h_{K-1}\}$

$x_1 = x_0 - \frac{h_0}{1+m}\nabla f(x_0)$

for $t = 1, 2, \dots$ **do**

$$x_{t+1} = x_t - h_{\mathrm{mod}(t,K)}\nabla f(x_t) + m(x_t - x_{t-1})$$

end

Our **main contribution** provides a positive answer to this question and, more importantly, *quantifies* the speedup under different assumptions. In particular, we show that for quadratic problems, whenever Hessian's spectrum belongs to two or more disjoint intervals, the heavy ball method with cyclical step-sizes achieves a faster worst-case convergence rate. Recent works have shown that this assumption on the spectrum is quite natural and occurs in many machine learning problems, including deep neural networks [103, 90, 37, 91]. The concurrent work of Oymak [88] analyzes gradient descent (without momentum) under a similar assumption.

Contributions: In [C9],

- We provide a **tight convergence rate analysis** of the cyclical heavy ball method (see Theorem 5.2 and corollary 5.3 for two step-sizes, and Theorem 4.8 in [C9] for the general case). This analysis highlights a regime under which this method achieves a faster worst-case rate than the accelerated rate of heavy ball, a phenomenon we refer to as *super-acceleration*. We also extend the (local) convergence rate analysis results to non-quadratic objectives.
- As a byproduct of the convergence-rate analysis, we obtain an explicit expression for the **optimal parameters** in in the case of cycles of length two (Algorithm 4) and an implicit expression in terms of a system of K equations in the general case.
- **Numerical benchmarks** illustrate the improved convergence of the cyclical approach on 4 problems involving quadratic and logistic losses on both synthetic and a hand-written digits recognition dataset.

Notation and Problem Setting As throughout the chapter, we consider the problem of minimizing a quadratic function:

$$\min_{x\in\mathbb{R}^d} f(x)\,, \text{ with } \quad f \in \mathcal{C}_\Lambda, \tag{OPT}$$

where $\mathcal{C}_\Lambda$ is the class of quadratic functions with Hessian matrix H and whose Hessian spectrum $\mathrm{Sp}(H)$ is localized in $\Lambda \subseteq [\mu, L] \subseteq \mathbb{R}_{>0}$:

$$\mathcal{C}_\Lambda \triangleq \left\{ f(x) = (x - x_*)^\top \tfrac{H}{2}(x - x_*) + f_*,\ \mathrm{Sp}(H) \subseteq \Lambda \right\}$$

The condition $\Lambda \subseteq [\mu, L]$ implies all quadratic functions under consideration are L-smooth and μ-strongly convex. For this function class, we define κ, the (inverse) condition number, and ρ, the ratio between the center of Λ and its radius, as

$$\kappa \triangleq \frac{\mu}{L}, \qquad \rho \triangleq \frac{L+\mu}{L-\mu} = \left(\frac{1+\kappa}{1-\kappa}\right). \tag{5.7}$$

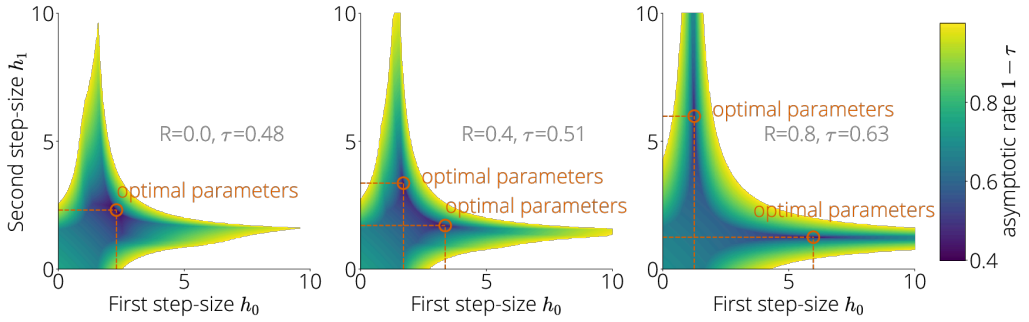


Figure 5.3: **Asymptotic rate of cyclical ($K = 2$) heavy ball** in terms of its step-sizes h_0, h_1 across 3 different values of the relative gap R. In the **left** plot, the relative gap is zero, and so the step-sizes with smallest rate coincide ($h_0 = h_1$). For non-zero values of R (**center and right**), the optimal method instead alternates between two *different* step-sizes. In all plots the momentum parameter m is set according to Algorithm 4.

Finally, for a method solving (OPT) that generates a sequence of iterates $\{x_t\}$, we define its worst-case rate r_t and its asymptotic rate factor τ as

$$r_t \triangleq \sup_{x_0 \in \mathbb{R}^d,\, f \in \mathcal{C}_\Lambda} \frac{\|x_t - x_*\|}{\|x_0 - x_*\|}, \quad 1 - \tau \triangleq \limsup_{t \to \infty} \sqrt[t]{r_t}\,. \tag{5.8}$$

5.2.1 Super-acceleration with Cyclical Step-sizes

In this section we develop one of our main contributions, a convergence rate analysis of the cyclical heavy ball method with cycles of length 2. This analysis crucially depends on the location of the Hessian's eigenvalues; we assume that these are contained in a set Λ that is the union of 2 intervals *of the same size*

$$\Lambda = [\boldsymbol{\mu}_1,\, \boldsymbol{L}_1] \cup [\boldsymbol{\mu}_2,\, \boldsymbol{L}_2]\,, \quad \boldsymbol{L}_1 - \boldsymbol{\mu}_1 = \boldsymbol{L}_2 - \boldsymbol{\mu}_2\,. \tag{5.9}$$

By symmetry, this set is alternatively described by

$$\mu \triangleq \boldsymbol{\mu}_1, \quad L \triangleq \boldsymbol{L}_2 \quad \text{and} \quad R \triangleq \frac{\boldsymbol{\mu}_2 - \boldsymbol{L}_1}{\boldsymbol{L}_2 - \boldsymbol{\mu}_1}\,, \tag{5.10}$$

where R is the relative length of the gap $\boldsymbol{\mu}_2 - \boldsymbol{L}_1$ with respect to the diameter $\boldsymbol{L}_2 - \boldsymbol{\mu}_1$ (see Figure 5.2). This parametrization is convenient since the relative gap plays a crucial role in our convergence analysis. Our results allow $R = 0$, therefore recovering the classical setting of Hessian eigenvalues contained in an interval.

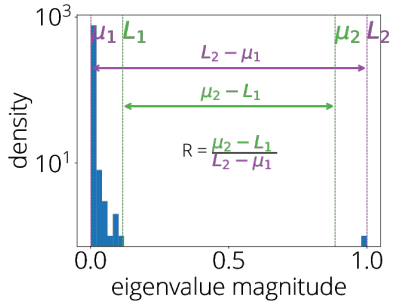


Figure 5.2: Hessian eigenvalue histogram for a quadratic objective on MNIST. The outlier eigenvalue at $\boldsymbol{L}_2$ generates a non-zero relative gap $R = 0.77$. In this case, the 2-cycle heavy ball method has a faster asymptotic rate than the single-cycle one (see Section 7).

Through a correspondence between optimization methods and polynomials, we can derive a worst-case analysis for the cyclical heavy ball method. The outcome of this analysis is in the following theorem, that provides the asymptotic convergence rate of Algorithm 3 for cycles of length two.

Theorem 5.2 (Rate factor of $\mathrm{HB}_2(h_0, h_1; m)$)**.** *Let $f \in \mathcal{C}_\Lambda$ and consider the cyclical heavy ball method with step-sizes h_0, h_1 and momentum parameter m. The asymptotic rate factor of*

Algorithm 3 with cycles of length two is

$$1-\tau = \begin{cases} \sqrt{m} & \text{if } \sigma_* \leqslant 1, \\ \sqrt{m}\left(\sigma_* + \sqrt{\sigma_*^2-1}\right)^{\frac{1}{2}} & \text{if } \sigma_* \in \left(1,\ \frac{1+m^2}{2m}\right), \\ \geqslant 1 \text{ (no convergence)} & \text{if } \sigma_* \geqslant \frac{1+m^2}{2m}, \end{cases}$$

$$\text{with} \quad \sigma_* = \max_{\lambda \in \left\{\mu_1, L_1, \mu_2, L_2, (1+m)\frac{h_0+h_1}{2h_0h_1}\right\} \cap \Lambda} |\sigma_2(\lambda)|$$

$$\text{and} \quad \sigma_2(\lambda) = 2\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right) - 1\,.$$

Optimal algorithm. The previous theorem gives the convergence rate for all triplets (h_0, h_1, m). This allows us for instance to map out the associated convergence rate for every pair of step-sizes. As we illustrate in Figure 5.3, as we increase the relative gap (R), the optimal step-sizes become further apart.

Another application of the previous theorem is to find the parameters that minimize the asymptotic convergence rate. The resulting momentum (m) and step-size parameters (h_0, h_1) are remarkably simple, and given by the expressions

$$m = \left(\frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\right)^2 \tag{5.11}$$

$$h_0 = \frac{1+m}{L_1} \qquad h_1 = \frac{1+m}{\mu_2}\,. \tag{5.12}$$

Being one of our main contributions, this algorithm is also described in pseudocode in Algorithm 4. By construction, this method has an *asymptotically optimal* convergence rate which we detail in the next Corollary:

Algorithm 4: Cyclical $(K=2)$ heavy ball with optimal parameters

Input: Initial iterate x_0, $\mu_1 < L_1 \leqslant \mu_2 < L_2$ (where $L_1 - \mu_1 = L_2 - \mu_2$)
Set: $\rho = \frac{L_2+\mu_1}{L_2-\mu_1}$, $R = \frac{\mu_2-L_1}{L_2-\mu_1}$,
$m = \left(\frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\right)^2$
$x_1 = x_0 - \frac{1}{L_1}\nabla f(x_0)$
for $t = 1, 2, \ldots$ **do**
  $h_t = \frac{1+m}{L_1}$ (if t is even), $\quad h_t = \frac{1+m}{\mu_2}$ (if t is odd)
  $x_{t+1} = x_t - h_t \nabla f(x_t) + m(x_t - x_{t-1})$
end

Corollary 5.3. *The non-asymptotic and asymptotic worst-case rates* $r_t^{\text{Alg. 2}}$ *and* $1-\tau^{\text{Alg. 2}}$ *of Algorithm 4 over* $\mathcal{C}_\Lambda$ *for even iteration number* t *are*

$$r_t^{\text{Alg. 2}} = \left(\frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\right)^t \left(1 + t\sqrt{\frac{\rho^2-1}{\rho^2-R^2}}\right),$$

$$1-\tau^{\text{Alg. 2}} = \frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\,.$$

Note that this result also holds if we swap the 2 step-sizes in Algorithm 4.

Eigengap and accelerated cyclical step-sizes. While Corollary 5.3 focuses on the optimal tuning of Algorithm 4, we also provide a general convergence analysis for non-optimal parameters [C9, Theorem D1]. In the case of existence of an eigengap, a range of cyclical step-sizes leads to an accelerated rate of convergence (compared to the optimal constant step-size strategy) and therefore, an inexact parameters search can lead to such an acceleration.

Comparison with Polyak Heavy Ball. In the absence of eigenvalue gap ($R = 0$ and $\Lambda = [\mu, L]$), Algorithm 4 reduces to Polyak heavy ball (PHB) [97]. Since the asymptotic rate of Algorithm 4 is monotonically decreasing in R, the convergence rate of the cyclical variant is always better than PHB. Furthermore, in the ill-conditioned regime (small κ), the comparison is particularly simple: the optimal 2-cycle algorithm has a $\sqrt{1-R^2}$ relative improvement over PHB, as provided by the next proposition. A more thorough comparison for different support sets Λ is discussed in Table 5.1.

Proposition 5.4. *Let $R \in [0, 1)$. The rate factors of respectively Algorithm 4 and PHB verify*

$$1 - \tau^{\text{Alg. 2}} \underset{\kappa \to 0}{=} 1 - \frac{2\sqrt{\kappa}}{\sqrt{1-R^2}} + o(\sqrt{\kappa}) \,, \qquad (5.13)$$
$$1 - \tau^{\text{PHB}} \underset{\kappa \to 0}{=} 1 - 2\sqrt{\kappa} + o(\sqrt{\kappa}) \,.$$

Relative gap R	Set Λ	Rate factor τ	Speedup τ/τ^{PHB}
$R \in [0, 1)$	$[\mu, \mu + \frac{1-R}{2}(L-\mu)] \cup [L - \frac{1-R}{2}(L-\mu), L]$	$\frac{2\sqrt{\kappa}}{\sqrt{1-R^2}}$	$(1-R^2)^{-\frac{1}{2}}$
$R = 1 - \sqrt{\kappa}/2$	$[\mu, \mu + \frac{\sqrt{\mu L}}{4}] \cup [L - \frac{\sqrt{\mu L}}{4}, L]$	$2\sqrt[4]{\kappa}$	$\kappa^{-\frac{1}{4}}$
$R = 1 - 2\gamma\kappa$	$[\mu, (1+\gamma)\mu] \cup [L - \gamma\mu, L]$	indep. of κ	$O(\kappa^{-\frac{1}{2}})$

Table 5.1: Case study of the convergence of Algorithm 4 as a function of R, in the regime $\kappa \to 0$. The **first line** corresponds to the regime where R is independent of κ, and we observe a constant gain w.r.t. PHB. The **second line** considers a setting in which R depends on $\sqrt{\kappa}$, that is, the two intervals in Λ are relatively small. The asymptotic rate reads $(1 - 2\sqrt[4]{\kappa})^t$, improving over the $(1 - 2\sqrt{\kappa})^t$ rate of Polyak Heavy ball, unimprovable when $R = 0$. Finally, in the **third line**, R depends on κ, the two intervals in Λ are so small that the convergence becomes $O(1)$, i.e., is independent of κ.

Extensions: in the referenced paper, we present a generic framework that allows designing optimal momentum and step-size cycles for given sets Λ and cycle length K, relying on the equioscillation property of polynomials. We provide convergence rate in the generalized case.

5.2.2 Experiments

We conclude by present an empirical comparison of the cyclical heavy ball method for different length cycles across 4 different settings. We consider two different problems,

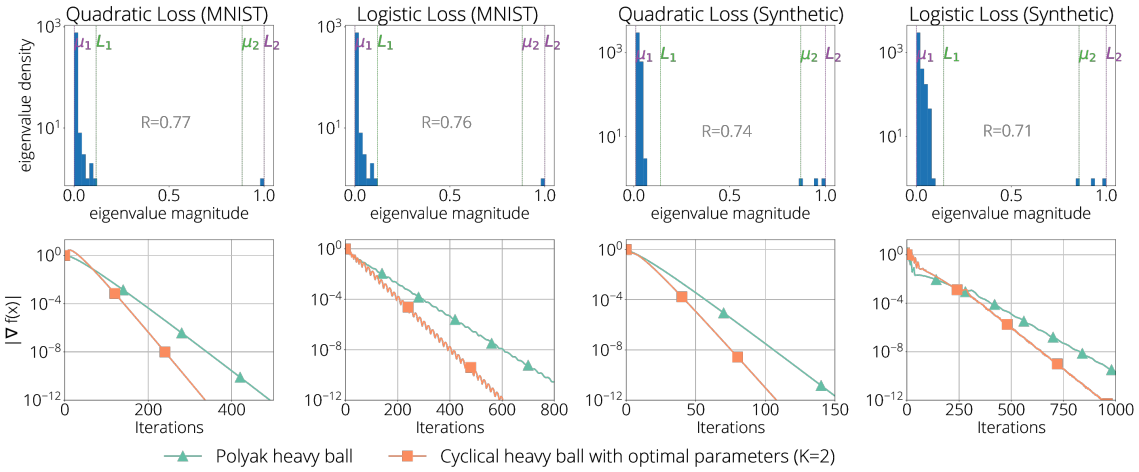


Figure 5.4: *Hessian Eigenvalue histogram (top row) and Benchmarks (bottom row)*. The **top row** shows the Hessian eigenvalue histogram at optimum for the 4 considered problems, together with the interval boundaries $\mu_1 < L_1 < \mu_2 < L_2$ for the two-interval split of the eigenvalue support described in Section 5.2.1. In all cases, there's a non-zero gap radius R. This is shown in the **bottom row**, where we compare the suboptimality in terms of gradient norm as a function of the number of iterations. As predicted by the theory, the non-zero gap radius translates into a faster convergence of the cyclical approach, compared to PHB in all cases. The improvement is observed on both quadratic and logistic regression problems, even through the theory for the latter is limited to *local* convergence.

quadratic and logistic regression, each applied on two datasets, the MNIST handwritten digits [67] and a synthetic dataset. The results of these experiments, together with a histogram of the Hessian's eigenvalues are presented in Figure 5.4 (see caption for a discussion).

Dataset description. The MNIST dataset consists of a data matrix A with 60000 images of handwritten digits each one with $28 \times 28 = 784$ pixels. The *synthetic* dataset is generated according to a spiked covariance model [56], which has been shown to be an accurate model of covariance matrices arising for instance in spectral clustering [15] and deep networks [93, 47]. In this model, the data matrix $A = XZ$ is generated from a $m \times n$ random Gaussian matrix X and an $m \times m$ deterministic matrix Z. In our case, we take $n = 1000, m = 1200$ and Z is the identity where the first three entries are multiplied by 100 (this will lead to three outlier eigenvalues). We also generate an n-dimensional target vector b as $b = Ax$ or $b = \text{sign}(Ax)$ for the quadratic and logistic problem respectively.

Objective function For each dataset, we consider a quadratic and a logistic regression problem, leading to 4 different problems. All problems are of the form $\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \ell(A_i^\top x, b_i) + \lambda \|x\|^2$, where ℓ is a quadratic or logistic loss, A is the data matrix and b are the target values. We set the regularization parameter to $\lambda = 10^{-3}\|A\|^2$. For logistic regression, since guarantees only hold at a neighborhood of the solution (even for the 1-cycle algorithm), we initialize the first iterate as the result of 100 iteration of gradient descent. In the case of logistic regression, the Hessian eigenvalues are computed at the optimum.

Conclusion This work is motivated by two recent observations from the optimization practice of machine learning. First, cyclical step-sizes have been shown to enjoy excellent empirical convergence [76, 110]. Second, *spectral gaps* are pervasive in the Hessian

spectrum of deep learning models [103, 90, 37, 91]. Based on the simpler context of quadratic convex minimization, we develop a convergence-rate analysis and optimal parameters for the heavy ball method with cyclical step-sizes. This analysis highlights the regimes under which cyclical step-sizes have faster rates than classical accelerated methods. Finally, we illustrate these findings through numerical benchmarks.

Bibliography

[1] Hadi Abbaszadehpeivasti, Etienne de Klerk, and Moslem Zamani. The exact worst-case convergence rate of the gradient method with fixed step lengths for L-smooth functions. *Optimization Letters*, 2021.

[2] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7564–7575. Curran Associates, Inc., 2018.

[3] Alham Fikri Aji and Kenneth Heafield. Sparse Communication for Distributed Gradient Descent. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445, 2017. doi: 10.18653/v1/D17-1045. arXiv: 1704.05021.

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. *Advances in Neural Information Processing Systems*, 30:1709–1720, 2017.

[5] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cedric Renggli. The Convergence of Sparsified Gradient Methods. *Advances in Neural Information Processing Systems*, 31:5973–5983, 2018.

[6] Robert B Ash. *Information theory*. Courier Corporation, 2012.

[7] Alexis Ayme, Claire Boyer, Aymeric Dieuleveut, and Erwan Scornet. Near-optimal rate of consistency for linear models with missing values. In *International Conference on Machine Learning*, pages 1211–1243. PMLR, 2022.

[8] Alexis Ayme, Claire Boyer, Aymeric Dieuleveut, and Erwan Scornet. Naive imputation implicitly regularizes high-dimensional linear models. *arXiv preprint arXiv:2301.13585*, 2023.

[9] Mathieu Barré, Adrien Taylor, and Alexandre d'Aspremont. Complexity guarantees for Polyak steps with momentum. In *Conference on Learning Theory (COLT)*, pages 452–478. PMLR, 2020.

[10] Léon Bottou. On-line learning and stochastic approximations. 1999. doi: 10.1017/CBO9780511569920.003.

[11] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2003.

[12] Rich Caruana, Thorsten Joachims, and Lars Backstrom. KDD-Cup 2004: results and analysis. *ACM SIGKDD Explorations Newsletter*, 6(2):95–108, December 2004. ISSN 1931-0145. doi: 10.1145/1046456.1046470.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199.

[14] Sebastien Colla and Julien M. Hendrickx. Automated worst-case performance analysis of decentralized gradient descent. In *Proceedings of the 60th Conference on Decision and Control (CDC)*, 2021.

[15] Romain Couillet and Florent Benaych-Georges. Kernel spectral clustering of large dimensional data. *Electronic Journal of Statistics*, 2016.

[16] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large Scale Distributed Deep Networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[17] Aaron Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[18] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Ann. Statist.*, 27(1):94–128, 1999.

[19] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Stat. Soc. B Met.*, 39(1):1–38, 1977.

[20] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research (JMLR)*, 17(1):2909–2913, 2016.

[21] Aymeric Dieuleveut and Francis Bach. Nonparametric stochastic approximation with large step-sizes. *The Annals of Statistics*, 44(4):1363–1399, 2016.

[22] Aymeric Dieuleveut and Kumar Kshitij Patel. Communication trade-offs for local-sgd with large step size. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Aymeric Dieuleveut, Nicolas Flammarion, and Francis Bach. Harder, better, faster, stronger convergence rates for least-squares regression. *The Journal of Machine Learning Research*, 18(1):3520–3570, 2017. Publisher: JMLR. org.

[24] Aymeric Dieuleveut, Alain Durmus, and Francis Bach. Bridging the Gap between Constant Step Size Stochastic Gradient Descent and Markov Chains. *arXiv:1707.06386 [math, stat]*, April 2018. arXiv: 1707.06386.

[25] Aymeric Dieuleveut, Gersende Fort, Eric Moulines, and Geneviève Robin. Federated-em with heterogeneity mitigation and variance reduction. *Advances in Neural Information Processing Systems*, 34:29553–29566, 2021.

[26] Aymeric Dieuleveut, Gersende Fort, Eric Moulines, and Wai Hoi-To. Stochastic approximation beyond gradient for signal processing and machine learning. *arXiv preprint*, 2023.

[27] Yoel Drori. *Contributions to the Complexity Analysis of Optimization Algorithms*. PhD thesis, Tel-Aviv University, 2014.

[28] Yoel Drori and Marc Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145(1):451–482, 2014.

[29] C. Fang, C.J. Li, Z. Lin, and T. Zhang. SPIDER: Near-Optimal Non-Convex Optimization via Stochastic Path-Integrated Differential Estimator. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 689–699. Curran Associates, Inc., 2018.

[30] Mahyar Fazlyab, Alejandro Ribeiro, Manfred Morari, and Victor M Preciado. Analysis of optimization algorithms via integral quadratic constraints: Nonstrongly convex problems. *SIAM Journal on Optimization*, 28(3):2654–2689, 2018.

[31] Gersende Fort, Eric Moulines, and Hoi-To Wai. A Stochastic Path Integral Differential EstimatoR Expectation Maximization Algorithm. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16972–16982. Curran Associates, Inc., 2020.

[32] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

[33] Sylvia Frühwirth-Schnatter, Gilles Celeux, and Christian P. Robert, editors. *Handbook of mixture analysis*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, Boca Raton, FL, 2019.

[34] Oran Gannot. A frequency-domain analysis of inexact gradient methods. *Mathematical Programming*, pages 1–42, 2021.

[35] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[36] S. Ghadimi and G. Lan. Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming. *SIAM J. Optim.*, 23(4):2341–2368, 2013.

[37] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning (ICML)*, 2019.

[38] E. Gorbunov, F. Hanzely, and P. Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 680–690. PMLR, 2020.

[39] Eduard Gorbunov, Dmitry Kovalev, Dmitry Makarenko, and Peter Richtarik. Linearly Converging Error Compensated SGD. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20889–20900. Curran Associates, Inc., 2020.

[40] Baptiste Goujaud, Céline Moucer, François Glineur, Julien Hendrickx, Adrien Taylor, and Aymeric Dieuleveut. Pepit: computer-assisted worst-case analyses of first-order optimization methods in python. *arXiv preprint arXiv:2201.04040*, 2022.

[41] Baptiste Goujaud, Damien Scieur, Aymeric Dieuleveut, Adrien B Taylor, and Fabian Pedregosa. Super-acceleration with cyclical step-sizes. In *International Conference on Artificial Intelligence and Statistics*, pages 3028–3065. PMLR, 2022.

[42] Baptiste Goujaud, Adrien Taylor, and Aymeric Dieuleveut. Optimal first-order methods for convex functions with a quadratic upper bound. *arXiv preprint arXiv:2205.15033*, 2022.

[43] Baptiste Goujaud, Adrien Taylor, and Aymeric Dieuleveut. Quadratic minimization: from conjugate gradient to an adaptive heavy-ball method with polyak step-sizes. *arXiv preprint arXiv:2210.06367*, 2022.

[44] Baptiste Goujaud, Aymeric Dieuleveut, and Adrien Taylor. Counter-examples in first-order optimization: a constructive approach. *arXiv preprint arXiv:2303.10503*, 2023.

[45] Robert M. Gower, Mathieu Blondel, Nidham Gazagnadou, and Fabian Pedregosa. Cutting some slack for SGD with Adaptive Polyak Stepsizes. *arXiv preprint arXiv:2202.12328*, 2022.

[46] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General Analysis and Improved Rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, May 2019. ISSN: 2640-3498.

[47] Diego Granziol, Xingchen Wan, Samuel Albanie, and Stephen Roberts. Explaining the Adaptive Generalisation Gap. *arXiv preprint arXiv:2011.08181*, 2020.

[48] Guoyong Gu and Junfeng Yang. Tight sublinear convergence rate of the proximal point algorithm for maximal monotone inclusion problems. *SIAM Journal on Optimization*, 30(3):1905–1921, 2020.

[49] Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, November 2018. ISSN 0927-0256. doi: 10.1016/j.commatsci.2018.07.052.

[50] Elad Hazan and Sham Kakade. Revisiting the Polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

[51] S. Horváth and P. Richtarik. A better alternative to error feedback for communication-efficient distributed learning. In *International Conference on Learning Representations*, 2021.

[52] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.

[53] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic Distributed Learning with Gradient Quantization and Variance Reduction. *arXiv:1904.05115 [math]*, April 2019. arXiv: 1904.05115.

[54] Bin Hu, Stephen Wright, and Laurent Lessard. Dissipativity theory for accelerating stochastic variance reduction: A unified analysis of svrg and katyusha using semidefinite programs. In *International Conference on Machine Learning (ICML)*, 2018.

[55] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and zhifeng Chen. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[56] Iain M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Annals of statistics*, 2001.

[57] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawit, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. *Advances and Open Problems in Federated Learning*. Now Foundations and Trends.

[58] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, May 2019. ISSN: 2640-3498.

[59] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/karimireddy20a.html.

[60] Sarit Khirirat, Sindri Magnússon, Arda Aytekin, and Mikael Johansson. Communication Efficient Sparsification for Large Scale Machine Learning. *arXiv:2003.06377 [math, stat]*, March 2020. arXiv: 2003.06377.

[61] Donghwan Kim. Accelerated proximal point method for maximally monotone operators. *Mathematical Programming*, pages 1–31, 2021.

[62] Donghwan Kim and Jeffrey A. Fessler. Optimized first-order methods for smooth convex minimization. *Mathematical Programming*, 159(1-2):81–107, 2016.

[63] Donghwan Kim and Jeffrey A. Fessler. Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions. *Journal of Optimization Theory and Applications*, 188(1):192–219, 2021.

[64] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv:1610.02527 [cs]*, October 2016. arXiv: 1610.02527.

[65] Frederik Kunstner, Raunak Kumar, and Mark Schmidt. Homeomorphic-invariance of em: Non-asymptotic convergence in kl divergence for exponential families via mirror descent. In *International Conference on Artificial Intelligence and Statistics*, pages 3295–3303. PMLR, 2021.

[66] K. Lange. *MM Optimization Algorithms*. SIAM-Society for Industrial and Applied Mathematics, 2016.

[67] Yann Le Cun, Corinna Cortes, and Chris Burges. MNIST handwritten digit database. *ATT Labs [Online]*, 2010.

[68] Louis Leconte, Aymeric Dieuleveut, Edouard Oyallon, Eric Moulines, and Gilles Pages. $\texttt{DoStoVoQ}$: Doubly Stochastic Voronoi Vector Quantization SGD for Federated Learning. May 2021.

[69] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[70] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*, OSDI'14, pages 583–598, USA, October 2014. USENIX Association. ISBN 978-1-931971-16-4.

[71] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the Convergence of FedAvg on Non-IID Data. October 2019.

[72] Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtarik. Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization. In *International Conference on Machine Learning*, pages 5895–5904. PMLR, November 2020. ISSN: 2640-3498.

[73] Felix Lieder. On the convergence rate of the Halpern-iteration. *Optimization Letters*, 15(2):405–418, 2021.

[74] Xiaorui Liu, Yao Li, Jiliang Tang, and Ming Yan. A Double Residual Compression Algorithm for Efficient Distributed Learning. In *International Conference on Artificial Intelligence and Statistics*, pages 133–143, June 2020. ISSN: 1938-7228 Section: Machine Learning.

[75] Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic Polyak step-size for SGD: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1306–1314. PMLR, 2021.

[76] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[77] Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I. Jordan. Perturbed Iterate Analysis for Asynchronous Stochastic Optimization. *arXiv:1507.06970 [cs, math, stat]*, March 2016. arXiv: 1507.06970.

[78] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[79] G.J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley series in probability and statistics. Wiley, 2008.

[80] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, April 2017. ISSN: 2640-3498.

[81] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed Learning with Compressed Gradient Differences. *arXiv:1901.09269 [cs, math, stat]*, June 2019. arXiv: 1901.09269.

[82] APS Mosek. The MOSEK optimization software. *Online at http://www.mosek.com*, 54, 2010.

[83] K.P. Murphy and S. J. Russell. Dynamic bayesian networks: representation, inference and learning. 2002.

[84] Arkadi S. Nemirovsky. Information-based complexity of linear operator equations. *Journal of Complexity*, 1992.

[85] Yurii Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2003.

[86] Maxence Noble, Aurélien Bellet, and Aymeric Dieuleveut. Differentially private federated learning on heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 10110–10145. PMLR, 2022.

[87] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, 1999.

[88] Samet Oymak. Super-convergence with an unstable learning rate. *arXiv preprint arXiv:2102.10734*, 2021.

[89] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

[90] Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with SGD training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

[91] Vardan Papyan. Measurements of Three-Level Hierarchical Structure in the Outliers in the Spectrum of Deepnet Hessians. In *International Conference on Machine Learning (ICML)*, 2019.

[92] Panagiotis Patrinos, Lorenzo Stella, and Alberto Bemporad. Douglas-Rachford splitting: Complexity estimates and accelerated variants. In *Proceedings of the 53rd Conference on Decision and Control (CDC)*, 2014.

[93] Jeffrey Pennington and Pratik Worah. Nonlinear random matrix theory for deep learning. In *Advances on Neural Information Processing Systems (NIPS)*, 2017.

[94] Scott Pesme, Aymeric Dieuleveut, and Nicolas Flammarion. On convergence-diagnostic based step sizes for stochastic gradient descent. In *International Conference on Machine Learning*, pages 7641–7651. PMLR, 2020.

[95] Constantin Philippenko and Aymeric Dieuleveut. Artemis: tight convergence guarantees for bidirectional compression in Federated Learning. *arXiv:2006.14591 [cs, stat]*, November 2020. arXiv: 2006.14591.

[96] Constantin Philippenko and Aymeric Dieuleveut. Preserved central model for faster bidirectional compression in distributed settings. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[97] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[98] Boris T. Polyak. Introduction to optimization. 1987. *Optimization Software, Inc, New York*, 1987.

[99] Ali Ramezani-Kebrya, Fartash Faghri, and Daniel M Roy. Nuqsgd: Improved communication efficiency for data-parallel sgd via nonuniform quantization. *arXiv preprint arXiv:1908.06077*, 2019.

[100] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031. PMLR, June 2020. ISSN: 2640-3498.

[101] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *Annals of Mathematical Statistics*, 22(3): 400–407, September 1951. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177729586. Number: 3 Publisher: Institute of Mathematical Statistics.

[102] Ernest K. Ryu, Adrien B. Taylor, Carolina Bergeling, and Pontus Giselsson. Operator splitting performance estimation: Tight contraction factors and optimal parameter selection. *SIAM Journal on Optimization*, 30(3):2251–2271, 2020.

[103] Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the Hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[104] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2019. ISSN 2162-2388. doi: 10.1109/TNNLS.2019.2944481. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[105] Kevin Scaman, Francis Bach, Sebastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal Algorithms for Non-Smooth Distributed Optimization in Networks. *Advances in Neural Information Processing Systems*, 31:2740–2749, 2018.

[106] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, March 2017. ISSN 1436-4646. doi: 10.1007/s10107-016-1030-6.

[107] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*. Citeseer, 2014.

[108] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[109] Sidak Pal Singh, Andreas Hug, Aymeric Dieuleveut, and Martin Jaggi. Context mover's distance & barycenters: Optimal transport of contexts for building representations. In *International Conference on Artificial Intelligence and Statistics*, pages 3437–3449. PMLR, 2020.

[110] Leslie N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017.

[111] Aude Sportisse, Claire Boyer, Aymeric Dieuleveut, and Julie Josse. Debiasing averaged stochastic gradient descent to handle missing values. *Advances in Neural Information Processing Systems*, 33:12957–12967, 2020.

[112] Sebastian U. Stich. Local SGD Converges Fast and Communicates Little. *arXiv:1805.09767 [cs, math]*, May 2019. arXiv: 1805.09767.

[113] Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for SGD with delayed gradients and compressed updates. *Journal of Machine Learning Research*, 21:1–36, 2020.

[114] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with Memory. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4447–4458. Curran Associates, Inc., 2018.

[115] Nikko Strom. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[116] Akhil Sundararajan, Bryan Van Scoy, and Laurent Lessard. Analysis and design of first-order distributed optimization algorithms over time-varying graphs. *IEEE Transactions on Control of Network Systems*, 7(4):1597–1608, 2020.

[117] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*, 2013.

[118] Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. DoubleSqueeze: Parallel Stochastic Gradient Descent with Double-pass Error-Compensated Compression. In *International Conference on Machine Learning*, pages 6155–6165. PMLR, May 2019. ISSN: 2640-3498.

[119] Adrien Taylor and Francis Bach. Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions. In *Proceedings of the 32nd Conference on Learning Theory (COLT)*, 2019.

[120] Adrien Taylor and Yoel Drori. An optimal gradient method for smooth strongly convex minimization. *arXiv:2101.09741*, 2021.

[121] 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.

[122] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. Performance estimation toolbox (PESTO): automated worst-case analysis of first-order optimization methods. In *Proceedings of the 56th Conference on Decision and Control (CDC)*, 2017.

[123] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, 161(1-2):307–345, 2017.

[124] Jean Ogier du Terrail, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold, Tanguy Marchand, Othmane Marfoq, Erum Mushtaq, et al. Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. *arXiv preprint arXiv:2210.04620*, 2022.

[125] Bryan Van Scoy, Randy A. Freeman, and Kevin M. Lynch. The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Systems Letters*, 2(1):49–54, 2017.

[126] C. Villani. *Optimal transport : old and new*. Grundlehren der mathematischen Wissenschaften. Springer, Berlin, 2009. ISBN 978-3-540-71049-3. URL http://opac.inria.fr/record=b1129524.

[127] Maxime Vono, Vincent Plassier, Alain Durmus, Aymeric Dieuleveut, and Eric Moulines. Qlsd: Quantised langevin stochastic dynamics for bayesian federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6459–6500. PMLR, 2022.

[128] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1509–1519. Curran Associates, Inc., 2017.

[129] Benedikt Wilbertz et al. Dual quantization for random walks with application to credit derivatives. *arXiv preprint arXiv:0910.5655*, 2009.

[130] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error Compensated Quantized SGD and its Applications to Large-scale Distributed Optimization. In *International Conference on Machine Learning*, pages 5325–5333. PMLR, July 2018. ISSN: 2640-3498.

[131] An Xu, Zhouyuan Huo, and Heng Huang. Training Faster with Compressed Gradient. *arXiv:2008.05823 [cs, stat]*, August 2020. arXiv: 2008.05823.

[132] Yue Yu, Jiaxiang Wu, and Longbo Huang. Double Quantization for Communication-Efficient Distributed Optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4438–4449. Curran Associates, Inc., 2019.

[133] Margaux Zaffran, Aymeric Dieuleveut, Julie Josse, and Yaniv Romano. Uncertainty quantification in presence of missing values. In *ICSDS 2022-IMS International Conference on Statistics and Data Science*, 2022.

[134] Margaux Zaffran, Olivier Féron, Yannig Goude, Julie Josse, and Aymeric Dieuleveut. Adaptive conformal predictions for time series. In *International Conference on Machine Learning*, pages 25834–25866. PMLR, 2022.

[135] Shuai Zheng, Ziyue Huang, and James Kwok. Communication-Efficient Distributed Blockwise Momentum SGD with Error-Feedback. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[136] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv:1606.06160 [cs]*, February 2018. arXiv: 1606.06160.

[137] D. L. Zhu and P. Marcotte. Co-Coercivity and Its Role In the Convergence of Iterative Schemes For Solving Variational Inequalities, March 1996.