Hessian-CoCoA: a general parallel and distributed framework for non-strongly convex regularizers

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A Distributed and Parallel Framework for Non-Strongly Convex Regularizers

Master Thesis

Matilde Gargiani

June 29th, 2017

Advisors: Prof. Dr. Thomas Hofmann, Prof. Dr. Martin Jaggi
Department of Computer Science, ETH Zürich

Abstract

The scale of modern datasets necessitates the development of efficient distributed and par- allel optimization methods for machine learning. We present a general-purpose framework for the distributed and parallel environments, Hessian-CoCoA, that has an efficient communication scheme and is applicable to a wide variety of problems in machine learning and signal pro- cessing. Hessian-CoCoA has been designed starting from an existing communication-efficient distributed framework, CoCoA, that has been shown to be competitive with the state-of-the- art distributed methods for non-strongly convex regularizers and non-smooth loss functions. In both the frameworks, instead of optimizing the original problem, the computation is dis- tributed by defining data-local subproblems that are solved independently in parallel. In particular, the main idea behind Hessian-CoCoA is to include more refined second order in- formation in the subproblems, providing a tighter and better local approximation than the one used by CoCoA. The new framework has markedly improved performance in terms of both convergence rate and total convergence time, as we will illustrate with an extensive set of experiments.

"If you would be a real seeker after truth, it is necessary that at least once in your life you doubt, as far as possible, all things."

— René Descartes

Acknowledgements

I would like to thank my supervisors Prof. Thomas Hofmann, Prof. Martin Jaggi, Andrew A. Bian and Celestine Dunner. I also would like to acknowledge my boyfriend Andrea and my dad Roberto, who have always supported me during these diffcult six months of work, encouraging and suggesting me.

Contents

Contents **v**

1 Introduction **1**
1.1 Regularized Loss Minimization 1
1.2 Algorithms 3
1.2.1 Gradient Descent 3
1.2.2 Newton-type Method 4
1.2.3 Proximal Gradient Descent 4
1.2.4 Proximal Newton-type Method 5
1.2.5 Coordinate Descent 5
1.2.6 (Block) Coordinate Gradient Descent Method 6
1.3 Primal-Dual Problems 6
1.3.1 Convex Conjugate 7
1.3.2 Fenchel-Rockafellar Duality 8
1.4 Definitions and Lemmas 9

2 CoCoA **13**
2.1 Setting 13
2.1.1 Primal-Dual Setting 13
2.1.2 Assumptions 14
2.2 Algorithmic Framework 16
2.2.1 Data Partitioning 16
2.2.2 Data-Local Subproblems 16
2.2.3 Reusability of Single Machines Solvers 18
2.2.4 Bounded Support Modification 18
2.3 Convergence Analysis 19
2.3.1 σ'_C Parameter 20
2.3.2 Lemmas and Theorem of Convergence 20

3 Hessian-CoCoA **27**

3.1 Data-Local Subproblems 27
3.2 Examples 29
3.3 Convergence Analysis 32
3.3.1 σ'_{HC} Parameter 32
3.3.2 Lemmas and Theorem of Convergence 35
3.3.3 Hessian-CoCoA and CoCoA Comparison for Logistic Regression 39
3.4 Another Perspective on CoCoA 40

4 Experiments **43**
4.1 Running Example 43
4.1.1 Primal Formulation 43
4.1.2 Dual Formulation 44
4.2 Local Solvers 44
4.3 Results 45

5 Parallel Framework **55**
5.1 State-of-the-Art 56
5.1.1 Stochastic CDN 56
5.1.2 Shotgun-CDN 57
5.2 Parallel Diagonal Hessian-CoCoA 59
5.3 Convergence Analysis 60
5.3.1 σ'_C Parameter 60
5.3.2 σ'_{HC} Parameter 61
5.3.3 Runtime Guarantee 61
5.4 Experiments 64

6 Conclusion and Future Directions **69**
6.1 Parallel Diagonal Hessian-CoCoA with Blocks 69
6.1.1 Block Spectral Radius 70
6.1.2 Feature Clustering 70
6.2 Mini-Batches Parallelization 71
6.2.1 Block-Structure 71

A Appendix **73**
A.1 Convex Conjugates 73
A.2 Convergence Result for Strongly Convex Regularizers 75
A.3 Runtime Guarantee for Stochastic CDN 78

Bibliography **85**

Chapter 1

Introduction

The volume of modern datasets is exploding at a rate never seen before: big data is arriving from multiple sources, such as social networks, smart phones, digital sensors, and it is changing so deeply our society that some experts are discussing about a big data revolution. Recent studies confirm that in the last decade more data has been created than in the entire previous history, and that, by the year 2020, about 1.7 megabytes of new information will be created every second for every human being on the planet. In order to analyse and extract meaningful information from this huge amount of data, scientists are called to redesign some of the classical machine learning algorithms, whose inherently sequential structure is not suited for this new setting. Therefore, parallel and distributed optimization methods for machine learning represent a fundamental topic for computer scientists, but also a really challenging one, both theoretically and in practice.

1.1 Regularized Loss Minimization

Many popular methods in machine learning and signal processing can be posed as unconstrained convex optimization problems of the following form, also known as empirical risk minimization (ERM) form:

$$\min_{\mathbf{u}\in\mathbb{R}^n} \quad \sum_{j=1}^{d} \ell_j(\mathbf{y}_j^\top \mathbf{u}) + r(\mathbf{u})\,, \tag{1.1}$$

where $\ell = \sum_j \ell_j$ is an empirical loss over the data, which depends on the specific application, and r is a regularizer.

Regularizers are generally used in machine learning to explicitly control the model complexity and, consequently, to avoid overfitting. The regularizers considered in our analysis are either of the form $r(\mathbf{u}) = \lambda\|\mathbf{u}\|_1$, or

$r(\mathbf{u}) = \frac{\lambda}{2}\|\mathbf{u}\|_2^2$, or, as in the elastic net case, a combination of the two of them, $r(\mathbf{u}) = \lambda(\frac{\eta}{2}\|\mathbf{u}\|_2^2 + \eta\|\mathbf{u}\|_1)$. For large data sets, the first choice is generally preferred, since a 1-norm regularizer induces sparsity in the solution, and, therefore, it simplifies the model and can be used for variables selection. On the other side, the resulting optimization problem is much more complicated to solve than the one resulting from the choice of a smooth norm regularizer, since it is not differentiable, and, consequently, the classical methods for smooth optimization can not be directly applied. To better illustrate the ERM structure, we consider some of the most famous applications where this optimization structure is used.

Lasso

$$\min_{\mathbf{u}\in\mathbb{R}^n} \quad \frac{1}{2}\|A\mathbf{u} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{u}\|_1\,, \tag{1.2}$$

where $A := [\mathbf{x}_1; \ldots; \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, is a data matrix with column vectors $\mathbf{x}_i \in \mathbb{R}^d$, $i \in [n]$, and $\mathbf{b} \in \mathbb{R}^d$, $\lambda \geq 0$.

Elastic Net Regression

$$\min_{\mathbf{u}\in\mathbb{R}^n} \quad \frac{1}{2}\|A\mathbf{u} - \mathbf{b}\|_2^2 + \sum_{i=1}^{n} \lambda(\frac{\eta}{2}(u_i)^2 + (1-\eta)|u_i|)\,, \tag{1.3}$$

where $A := [\mathbf{x}_1; \ldots; \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, is a data matrix with column vectors $\mathbf{x}_i \in \mathbb{R}^d$, $i \in [n]$, and $\mathbf{b} \in \mathbb{R}^d$, $\lambda \geq 0$, $\eta \geq 0$.

ℓ_2-Logistic Regression

$$\min_{\mathbf{u}\in\mathbb{R}^n} \quad \sum_{j=1}^{d} \log(1 + \exp(-b_j \mathbf{y}_j^\top \mathbf{u})) + \lambda\|\mathbf{u}\|_2^2\,, \tag{1.4}$$

where $A := [\mathbf{y}_1^\top, \ldots, \mathbf{y}_d^\top]$, is the data matrix with rows $\mathbf{y}_j \in \mathbb{R}^n$, $\forall j \in [d]$, and $\mathbf{b} \in \mathbb{R}^d$ with each element $b_i \in \{1, -1\}$, and $\lambda \geq 0$.

ℓ_1-Logistic Regression

$$\min_{\mathbf{u}\in\mathbb{R}^n} \quad \sum_{j=1}^{d} \log(1 + \exp(-b_j \mathbf{y}_j^\top \mathbf{u})) + \lambda\|\mathbf{u}\|_1\,, \tag{1.5}$$

where $A := [\mathbf{y}_1^\top, \ldots, \mathbf{y}_d^\top]$, is the data matrix with rows $\mathbf{y}_j \in \mathbb{R}^n$, $\forall j \in [d]$, and $\mathbf{b} \in \mathbb{R}^d$ with each element $b_i \in \{1, -1\}$, and $\lambda \geq 0$.

ℓ_2-Support Vector Machine

$$\min_{\mathbf{u}\in\mathbb{R}^n} \quad \sum_{j=1}^{d} \max\{0, 1 - b_j \mathbf{y}_j^\top \mathbf{u}\} + \frac{\lambda}{2}\|\mathbf{u}\|_2^2 , \tag{1.6}$$

where $A := [\mathbf{y}_1^\top, \dots, \mathbf{y}_d^\top]$, is the data matrix with rows $\mathbf{y}_j \in \mathbb{R}^n$, $\forall j \in [d]$, and $\mathbf{b} \in \mathbb{R}^d$ with each element $b_i \in \{1, -1\}$, and $\lambda \geq 0$.

ℓ_1-Support Vector Machine

$$\min_{\mathbf{u}\in\mathbb{R}^n} \quad \sum_{j=1}^{d} \max\{0, 1 - b_j \mathbf{y}_j^\top \mathbf{u}\} + \lambda\|\mathbf{u}\|_1 , \tag{1.7}$$

where $A := [\mathbf{y}_1^\top, \dots, \mathbf{y}_d^\top]$, is the data matrix with rows $\mathbf{y}_j \in \mathbb{R}^n$, $\forall j \in [d]$, and $\mathbf{b} \in \mathbb{R}^d$ with each element $b_i \in \{1, -1\}$, and $\lambda \geq 0$.

1.2 Algorithms

In this section, some of the major algorithms for smooth and non-smooth optimization are presented, whose structure is inherently sequential. We will not cover all the details, because it would exceed from the scope of the paragraph, which is intended to be just a brief overview over these methods.

1.2.1 Gradient Descent

Consider the following framework:

$$\min_{\mathbf{u}\in\mathbb{R}^n} f(\mathbf{u}) , \tag{1.8}$$

where the function f is smooth. The basic principle of the gradient descent method, see [9], is to minimize (1.8) by following the direction of the negative gradient, with a step length $\gamma \geq 0$:

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} - \gamma \nabla f(\mathbf{u}^{(t)}) .$$

There exists many versions of this method: the one exposed is known as standard, or batch, gradient method. Other commonly applied verions in large scale applications are stochastic gradient and mini-batch gradient descent. In the stochastic version, also called on-line gradient descent, the true gradient of f is approximated with the gradient at a single point i, that is chosen uniformly at random among the set of all the possible points. A compromise between computing the true gradient and the gradient at a single point is represented by the mini-batch version, since it requires the computation of the gradient against more than just a single point at each step.

1.2.2 Newton-type Method

Given (1.8), where f is assumed to be twice continuously differentiable, and given a matrix $\tilde{H}_t > 0$, with $\tilde{H}_t \in \mathbb{R}^{n \times n}$, one iteration of the Newton-type method (see [9] for more details) has the following structure:

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} - \gamma(\tilde{H}_t)^{-1}\nabla f(\mathbf{u}^{(t)})\,,$$

where $\gamma \geq 0$ is a chosen step length. The direction $\mathbf{d} = (\tilde{H}_t)^{-1}\nabla f(\mathbf{u}^{(t)})$ is called Newton-type step and it comes from the minimization of the following local quadratic approximation of f:

$$m_t(\mathbf{u}^{(t)} + \mathbf{d}) = f(\mathbf{u}^{(t)}) + \nabla f(\mathbf{u}^{(t)})^\top \mathbf{d} + \frac{1}{2\gamma}\mathbf{d}^\top \tilde{H}_t \mathbf{d}\,.$$

Given that $\tilde{H}_t > 0$, we have that $\mathbf{d}$ is always a descent direction. Regarding the local converge rate, the performances depends on $\tilde{H}_t$ and how close it is to the real Hessian matrix of f. If $\tilde{H}_t = \nabla^2 f(\mathbf{u}^{(t)})$, the algorithm corrisponds to the exact Newton method, therefore we can enojoy a quadratic local convergence rate. On the other hand, in the case where $\tilde{H}_t = I$, we obtain the over exposed gradient descent method.
Generally, the local convergence rate depends on the choice of $\tilde{H}_t$, and it goes from a quadratic rate for the exact Newton method, down to a linear rate for less refined approximation of $\nabla^2 f(\mathbf{u})$.

1.2.3 Proximal Gradient Descent

Given a composite function:

$$\min_{\mathbf{u} \in \mathbb{R}^n} f(\mathbf{u}) + r(\mathbf{u})\,, \tag{1.9}$$

where f is smooth and convex, and r is convex but not necessarily smooth, the gradient method can not be directly applied anymore, since globally the objective is non-smooth. In order to account for the non-smoothness of the r function, the proximal operator is introduced. The resulting algorithm is indeed called proximal gradient method and it updates the variables as follows:

$$\begin{aligned}\mathbf{u}^{(t+1)} &= \mathrm{prox}_{\gamma r}(\mathbf{u}^{(t)} - \gamma \nabla f(\mathbf{u}^{(t)})) \\ &= \mathbf{u}^{(t)} + \arg\min_{\mathbf{d} \in \mathbb{R}^n} \left\{ \nabla f(\mathbf{u}^{(t)})^\top \mathbf{d} + \frac{1}{2\gamma}\|\mathbf{d}\|_2^2 + r(\mathbf{u}^{(t)} + \mathbf{d}) \right\},\end{aligned}$$

where $\gamma \geq 0$ is a chosen step length.

The proximal gradient method can be interpreted as a generalization to non-smooth cases of its smooth version, the gradient method. Proximal gradient descent enjoys the same convergence rate as its fully smooth version, but the complexity of each iteration is now determined by the proximal operator of r. In the case of $r(\mathbf{u}) = \lambda\|\mathbf{u}\|_1$ the proximal step reduces to a soft threshold operator, which is not computationally expensive:

$$\mathbf{u}^{(t+1)} = \mathcal{S}_{\gamma\lambda}(\mathbf{u}^{(t)} - \gamma\nabla f(\mathbf{u}^{(t)}))\,. \tag{1.10}$$

1.2.4 Proximal Newton-type Method

The main idea behind the proximal gradient method is to iteratively minimize a quadratic expansion of the smooth component f, plus the original r. The quadratic approximation involved approximates the Hessian matrix of f by a scaled version of the identity $\frac{1}{\gamma}I$. As in the correspondent smooth version, the fundamental difference that leads from a proximal gradient to a proximal Newton method consists in replacing the scaled identity with a more refined local approximation of the Hessian matrix of the function f. These considerations lead to the proximal Newton method, which involves the use of a scaled proximal mapping. Let $\tilde{H}_t > 0$ be the local approximation of the Hessian matrix, and $\gamma > 0$ the step length, then the proximal Newton iteration consists of the following update:

$$\begin{aligned}\mathbf{u}^{(t+1)} &= \operatorname{prox}_{\frac{1}{\gamma}\tilde{H}_t}(\mathbf{u}^{(t)} - \gamma(\tilde{H}_t)^{-1}\nabla f(\mathbf{u}^{(t)})) \\ &= \mathbf{u}^{(t)} + \arg\min_{\mathbf{d}\in\mathbb{R}^n}\left\{\nabla f(\mathbf{u}^{(t)})^\top\mathbf{d} + \frac{1}{2\gamma}\mathbf{d}^\top\tilde{H}_t\mathbf{d} + r(\mathbf{u}^{(t)}+\mathbf{d})\right\}.\end{aligned}$$

1.2.5 Coordinate Descent

The growing interest towards Coordinate Descent (CD) algorithms (see [14]) is motivated by their competitive performance when applied to large scale machine learning problems. Indeed, despite their simplicity, these methods have shown to be very effective. The basic principle of CD methods is to solve an optimization problem by addressing a squence of simpler optimization problems, obtained by successively optimizing along coordinate directions (single-CD) or coordinate hyperplanes (block-CD). Many variants have been introduced, which differ from each others by the selection criteria of the optimization variables: one of the most famous in this perspective is stochastic-CD, where, at the beginning of each iteration, the coordinates, or block of coordinates, to be optimized is chosen uniformly at random from the set of all coordinates. Other more sophisticated selection criteria have

been explored as well in the literature (i.e. see [4]). In this section, we will focus on the simplest formulation of this class of algorithms, single-CD, where the coordinates are updated in a sequential fashion.

CD-Algorithm

Given a minimization over a multivariable function, $\min_{\mathbf{u}\in\mathbb{R}^n} f(\mathbf{u})$, one iteration of the single-CD procedure consists in iteratively updating the following one dimensional subproblems in order:

$$\begin{aligned}
\mathbf{u}_1^{(t+1)} &= \arg\min_{u\in\mathbb{R}} f(u; u_2^{(t)} \dots; u_n^{(t)}) \,, \\
\mathbf{u}_2^{(t+1)} &= \arg\min_{u\in\mathbb{R}} f(u_1^{(t+1)}; u; u_3^{(t)} \dots; u_n^{(t)}) \,, \\
&\vdots \\
\mathbf{u}_n^{(t+1)} &= \arg\min_{u\in\mathbb{R}} f(u_1^{(t+1)}; \dots; u_{n-1}^{(t+1)}; u) \,.
\end{aligned}$$

1.2.6 (Block) Coordinate Gradient Descent Method

Given (1.9) as operative framework, (block) coordinate gradient descent method, as described in the paper [10], comes from a combination of proximal Newton-type method and coordinate descent: indeed, it is based on a local quadratic approximation of the smooth component f, plus the original r, and the basic principle of coordinate descent is used to address the resulting optimization problem: not all the variables are optimized in one turn, but only a subset $\mathcal{J}$. More precisely, given a nonempty index subset $\mathcal{J} \subseteq \{1, \dots, n\}$ and a matrix $\tilde{H}_t > 0$ (approximating the Hessian of f), at iteration t, the (block) coordinate gradient descent method performs the following update:

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} + \arg\min_{\mathbf{d}\in\mathbb{R}^n} \left\{ \nabla f(\mathbf{u}^{(t)})^\top \mathbf{d} + \frac{1}{2\gamma} \mathbf{d}^\top \tilde{H}_t \mathbf{d} + r(\mathbf{u}^{(t)} + \mathbf{d}) |\ \ d_j = 0 \,, \forall j \notin \mathcal{J} \right\} , \tag{1.11}$$

where $\gamma \geq 0$ is the step length.

If $\tilde{H}_t$ is a (block) diagonal matrix and r is (block) separable, then (1.11) decomposes into independent subproblems that can be solved in parallel.

1.3 Primal-Dual Problems

This section is dedicated to a short analysis of one of the main preliminary concepts for this work: primal-dual relation for ERM problems. Often in the literature these problems have been addressed by dual methods, i.e.

stochastic dual coordinate ascent [12]: the main idea behind these methods is to optimize the dual formulation instead of the primal one, and to derive the primal optimal value and the primal optimal solution from the corresponding dual ones. The reason why this is possible is a well defined relation between primal and dual problems. In particular, we will focus on the Fenchel-Rockafellar duality, and we will show in Section 1.3.2 how, for ERM problems, it can be derived from the Lagrange dual formulation.

1.3.1 Convex Conjugate

An important concept that will be used throughout this work, is the convex conjugate of a function f.

Definition 1.1 *The convex conjugate of a function $f := \mathbb{R}^n \to \mathbb{R}$, is defined as*

$$f^*(\mathbf{v}) = \max_{\mathbf{u}\in\mathbb{R}^n} \mathbf{v}^\top \mathbf{u} - f(\mathbf{u})\,.$$

Lemma 1.2 *(Fenchel-Young Inequality) Given a function $f : \mathbb{R}^d \to \mathbb{R}$ and its convex conjugate f^*, it holds that*

$$\mathbf{u}^\top \mathbf{v} \leq f(\mathbf{u}) + f^*(\mathbf{v}) \quad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^d\,. \tag{1.12}$$

Lemma 1.3 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuous proper convex function and f^* its conjugate. Then,*

$$\mathbf{v} \in \delta f(\mathbf{u}) \implies f(\mathbf{u}) + f^*(\mathbf{v}) = \mathbf{u}^\top \mathbf{v}\,, \tag{1.13}$$

$$\mathbf{u} \in \delta f^*(\mathbf{v}) \implies f(\mathbf{u}) + f^*(\mathbf{v}) = \mathbf{u}^\top \mathbf{v}\,. \tag{1.14}$$

Proof We will prove only (1.13): since (1.14) can be easily derived by considering that $f = (f^*)^*$, given that f is a proper and continuous convex function. First, by Fenchel-Young inequality (1.12), we have that $f(\mathbf{u}) + f^*(\mathbf{v}) \geq \mathbf{u}^\top \mathbf{v}$. We thus only need to prove the reverse inequality. Since, by definition of subgradient $\mathbf{v} \in \delta f(\mathbf{u})$, we have that

$$f(\mathbf{z}) \geq f(\mathbf{v}) + (\mathbf{z} - \mathbf{v})^\top \mathbf{u} \quad \forall \mathbf{z} \in \mathbb{R}^d\,,$$

we can rewrite the definition of the convex conjugate as follows

$$\begin{aligned} f(\mathbf{u}) + f^*(\mathbf{v}) &= f(\mathbf{u}) + \sup_{\mathbf{z}} \Big[\mathbf{z}^\top \mathbf{v} - f(\mathbf{z})\Big] \\ &\leq f(\mathbf{u}) + \sup_{\mathbf{z}} \Big[\mathbf{z}^\top \mathbf{v} - (f(\mathbf{u}) + (\mathbf{z} - \mathbf{u})^\top \mathbf{v})\Big] \\ &= \sup_{\mathbf{z}} \mathbf{u}^\top \mathbf{v} = \mathbf{u}^\top \mathbf{v}\,. \end{aligned}$$

□

1.3.2 Fenchel-Rockafellar Duality

Given $A \in \mathbb{R}^{d \times n}$, with rows $\mathbf{y}_j \; \forall j \in \{1, \ldots, d\}$, we consider the ERM problem formulation

$$\min_{\mathbf{u} \in \mathbb{R}^n} \quad [P(\mathbf{u}) = \ell(A\mathbf{u}) + r(\mathbf{u})] \,, \tag{1.15}$$

and, by definition, its Fenchel-Rockafellar dual is given by:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \quad \left[D(\mathbf{w}) = -\ell^*(\mathbf{w}) - r^*(-A^\top \mathbf{w})\right] . \tag{1.16}$$

This can be directly derived from the Lagrange formulation. Indeed, starting from (1.15), and introducing the variable substitution $\mathbf{v} = A\mathbf{u}$, the problem becomes:

$$\min_{\substack{\mathbf{v} \in \mathbf{R}^d \\ \mathbf{u} \in \mathbb{R}^n}} \quad \ell(\mathbf{v}) + r(\mathbf{u}) \,, \text{ such that } \mathbf{v} = A\mathbf{u} \,.$$

By introducing the Lagrange multipliers $\mathbf{w} \in \mathbb{R}^d$ and dualizing the constraints, we obtain the Lagrangian:

$$L(\mathbf{u}, \mathbf{v}, \mathbf{w}) = \ell(\mathbf{v}) + r(\mathbf{u}) + \mathbf{w}^\top (A\mathbf{u} - \mathbf{v}) \,. \tag{1.17}$$

The Lagrangian dual formulation comes from taking the infimum with respect to both the primal variables $\mathbf{u}$ and $\mathbf{v}$:

$$\begin{aligned} \inf_{\mathbf{u},\mathbf{v}} \quad L(\mathbf{u}, \mathbf{v}, \mathbf{w}) &= \inf_{\mathbf{v}} \quad \{\ell(\mathbf{v}) - \mathbf{w}^\top \mathbf{v}\} + \inf_{\mathbf{u}} \quad \{r(\mathbf{u}) + \mathbf{w}^\top A\mathbf{u}\} \\ &= -\sup_{\mathbf{v}} \quad \{\mathbf{w}^\top \mathbf{v} - \ell(\mathbf{v})\} - \sup_{\mathbf{u}} \quad \{(-\mathbf{w}^\top A)\mathbf{u} - r(\mathbf{u})\} \\ &= -\ell^*(\mathbf{w}) - r^*(-A^\top \mathbf{w}) \,. \end{aligned}$$

In addition, by applying the first-order optimality conditions, we derive the following mapping between primal variable $\mathbf{u}$ and dual variable $\mathbf{w}$:

$$\mathbf{w} = \mathbf{w}(\mathbf{u}) := \delta\ell(A\mathbf{u}) \,, \tag{1.18}$$

that reduces to $\mathbf{w} = \nabla\ell(A\mathbf{u})$ if ℓ is smooth. Finally, the distance between primal and dual objective function is called duality gap $G(\mathbf{u})$ and it is always a non-negative quantity:

$$G(\mathbf{u}) := P(\mathbf{u}) - (-D(\mathbf{w}(\mathbf{u}))) \geq 0 \,, \tag{1.19}$$

where the equality holds at the optimum. The duality gap provides an upper bound on the optimization error, since, from what has been derived, it is clear that the distance bewteen $P(\mathbf{u})$ and $P(\mathbf{u}^*)$ is always upper bounded by the distance between the two objective values $P(\mathbf{u})$ and $D(\mathbf{w})$.

1.4 Definitions and Lemmas

In this section some of the main definitions and lemmas that are used in our analysis are exposed.

Definition 1.4 *(Convex Function). A real valued-differentiable function $h : \mathbb{R}^m \to \mathbb{R}$ is convex iff $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ and $\theta \in [0, 1]$*

$$h(\theta\mathbf{u} + (1-\theta)\mathbf{v}) \leq \theta h(\mathbf{u}) + (1-\theta)h(\mathbf{v})\,. \tag{1.20}$$

Definition 1.5 *(Lower Bounded Hessian Matrix). A real-valued twice differentiable function $h : \mathbb{R}^m \to \mathbb{R}$ is convex iff $\forall \mathbf{u} \in \mathbb{R}^m$*

$$\nabla^2 h(\mathbf{u}) \geq 0\,. \tag{1.21}$$

Definition 1.6 *(L-Lipschitz Continuity). A function $h : \mathbb{R}^m \to \mathbb{R}$ is L-Lipschitz continuous iff $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, there exists $L \geq 0$ such that*

$$|h(\mathbf{u}) - h(\mathbf{v})| \leq L\|\mathbf{u} - \mathbf{v}\|\,. \tag{1.22}$$

Definition 1.7 *(L-Bounded Support). A function $h : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ has L-bounded support with $L \geq 0$ iff its effective domain is bounded by L, i.e.*

$$h(\mathbf{u}) < +\infty \implies \|\mathbf{u}\| \leq L\,. \tag{1.23}$$

Lemma 1.8 *(Duality between Lipschitzness and L-Bounded Support) Given a proper convex function $f : \mathbb{R}^d \to \mathbb{R}$, it holds that f is L-Lipschitz if and only if its convex conjugate f^* has L-bounded support.*

Lemma 1.9 *(L-Bounded Derivative). Let $h : \mathbb{R}^m \to \mathbb{R}$ be continuous and differentiable in all the directions m. Then, h is Lipschitz continuous iff $\exists L \geq 0$ such that $\forall \mathbf{u} \in \mathbb{R}^m$, $\|\nabla h(\mathbf{u})\| \leq L$.*

Proof The proof follows directly from the definition of derivative.

- $(\Longleftarrow)$ trivially follows from the definition of derivative.
- $(\Longrightarrow)$ h is L Lipschitz by assumption. It follows that all the restrictions of h to a line parallel to a coordinate axis are L-Lipschitz. Without loss of generality, we assume $m = 2$

$$\begin{aligned}
|h(\mathbf{u}) - h(\mathbf{v})| &= |h(u_1, u_2) - h(v_1, v_2)| \\
&= |h(u_1, u_2) - h(u_1, v_2) + h(u_1, v_2) - h(v_1, v_2)| \\
&\leq |h(u_1, u_2) - h(u_1, v_2)| + |h(u_1, v_2) - h(v_1, v_2)| \\
&\leq L|u_2 - v_2| + L|u_1 - v_1|\,.
\end{aligned}$$

From that claim and by applying the definition of derivative along a direction

$$\begin{aligned}\nabla h(u_1, u_2) &= \lim_{h\to 0} \quad \frac{|h(u_1, u_2+h) - h(u_1, u_2)|}{h} \\ &\leq \lim_{h\to 0} \quad \frac{L|u_2 + h - u_2|}{h} \\ &\leq L\,. \end{aligned}$$

□

Definition 1.10 *($(\frac{1}{\mu})$-Smoothness). A function $h : \mathbb{R}^m \to \mathbb{R}$ is $(\frac{1}{\mu})$-smooth if it is differentiable and its derivative is $(\frac{1}{\mu})$-Lipschitz continuous, or equivalently*

$$h(\mathbf{u}) \leq h(\mathbf{v}) + \langle \nabla h(\mathbf{v}), \mathbf{u} - \mathbf{v}\rangle + \frac{1}{2\mu}\|\mathbf{u} - \mathbf{v}\|^2 \quad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^m. \tag{1.24}$$

Lemma 1.11 *($(\frac{1}{\mu})$-Bounded Hessian Matrix). Given a convex and twice differentiable function $h : \mathbb{R}^m \to \mathbb{R}$, it is $(\frac{1}{\mu})$-smooth iff $\nabla^2 h \leq \frac{1}{\mu} I$.*

Proof The following proof comes from previous considerations

$$\nabla^2 h \leq \frac{1}{\mu} I \iff \|\nabla^2 h\| \leq \frac{1}{\mu} \iff \|\nabla h(\mathbf{u}) - \nabla h(\mathbf{v})\| \leq \frac{1}{\mu}\|\mathbf{u} - \mathbf{v}\| \quad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^m\,.$$

□

Definition 1.12 *(μ-Strong Convexity). A function $h : \mathbb{R}^m \to \mathbb{R}$ is μ-strongly convex for $\mu \geq 0$ iff*

$$h(\mathbf{u}) \geq h(\mathbf{v}) + \langle \nabla h(\mathbf{v}), \mathbf{u} - \mathbf{v}\rangle + \frac{\mu}{2}\|\mathbf{u} - \mathbf{v}\|^2 \quad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^m\,. \tag{1.25}$$

Lemma 1.13 *(Duality between Smoothness and Strong Convexity) Given a closed convex function $f : \mathbb{R}^d \to \mathbb{R}$, it holds that f is μ- strongly convex w.r.t. the norm $\|\cdot\|$ if and only if its convex conjugate f^* is $(\frac{1}{\mu})$-smooth w.r.t. the dual norm $\|\cdot\|_*$.*

Definition 1.14 *(Compatible Norms). A matrix norm $\|\cdot\|_b$ on $\mathbb{R}^{m\times m}$ is called compatible with a vector norm $\|\cdot\|_a$ on $\mathbb{R}^m$ if $\forall A \in \mathbb{R}^{m\times m}$, $\mathbf{u} \in \mathbb{R}^m$*

$$\|A\mathbf{u}\|_a \leq \|A\|_b \|\mathbf{u}\|_a\,. \tag{1.26}$$

Definition 1.15 *(Induced Norms). Given a vector norm $\|\cdot\|$ on $\mathbb{R}^m$, the matrix norm $\|\cdot\|$ on $\mathbb{R}^{m\times m}$ induced by that vector norm is*

$$\|A\| = \sup_{\mathbf{u}\in\mathbb{R}^m, \mathbf{u}\neq 0} \frac{\|A\mathbf{u}\|}{\|\mathbf{u}\|}\,. \tag{1.27}$$

In addition, induced norms are compatible norms by definition.

Definition 1.16 *(Bounded Linear Maps). For every norm $\|\cdot\|$ on $\mathbb{R}^m$, and for every matrix $A \in \mathbb{R}^{m\times m}$, there is a real constant $C_A \geq 0$ such that*

$$\sup_{\mathbf{u}\in\mathbb{R}^m, \mathbf{u}\neq 0} \frac{\|A\mathbf{u}\|}{\|\mathbf{u}\|} \leq C_A \,. \tag{1.28}$$

Lemma 1.17 *Given $\ell = f \circ v$, where $f : \mathbb{R}^d \to \mathbb{R}$ is $(\frac{1}{\tau})$-smooth, and $v : \mathbb{R}^n \to \mathbb{R}^d$ is the affine map associated to the matrix $A \in \mathbb{R}^{d\times n}$, ℓ is β-smooth, and $\beta \propto \frac{1}{\tau}$.*

Proof $\forall \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in dom(\ell)$, and being $\mathbf{v}_1 = A\boldsymbol{\alpha}_1$, $\mathbf{v}_2 = A\boldsymbol{\alpha}_2$:

$$\begin{aligned}
\|\nabla\ell(\boldsymbol{\alpha}_1) - \nabla\ell(\boldsymbol{\alpha}_2)\| &= \|A(\nabla f(\mathbf{v}_1) - \nabla f(\mathbf{v}_2))\| \\
&\leq \|A\|\|f(\mathbf{v}_1) - f(\mathbf{v}_2)\| \\
&\leq \frac{1}{\tau}\|A\|\|\mathbf{v}_1 - \mathbf{v}_2\| \\
&\leq \frac{1}{\tau}\|A\|^2\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \\
&\leq \frac{C_A^2}{\tau}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| \,.
\end{aligned}$$

□

Definition 1.18 *(Univariate Self-Concordant Function). Let $k \geq 0$ and $\mathcal{D} \subseteq \mathbb{R}$. The univariate function $f : \mathcal{D} \to \mathbb{R}$ is called k-self concordant if*

$$\left|f'''(x)\right| \;\leq\; 2k\left(f''(x)\right)^{\frac{3}{2}} \quad \forall x \in \mathcal{D} \,. \tag{1.29}$$

For the multivariate case, the previous definition is valid if it can be applied on a restriction to an arbitrary line in the domain of the function.

Definition 1.19 *(Multivariate Self-Concordant Function). Let $k \geq 0$ and $\mathcal{D} \subseteq \mathbb{R}^d$. The function $f : \mathcal{D} \to \mathbb{R}$ is called k-self concordant if and only if $\varphi(\theta) := f(\mathbf{x} + \theta\mathbf{d})$ is k-self concordant at $\theta = 0$, for all $\mathbf{x} \in \mathcal{D}$ and $\mathbf{d} \in \mathbb{R}^d$*

$$\left|\varphi'''(0)\right| \;\leq\; 2k\varphi''(0)^{\frac{3}{2}} \quad \forall \mathbf{x} \in \mathcal{D} \,,\;\; \forall \mathbf{d} \in \mathbb{R}^d \,. \tag{1.30}$$

Some functions, such as the logistic regression loss, even if not self concordant, can be analyzed similarly to the class of self concordant functions, since their third derivative is bounded by a constant times the second derivative (for more details see [1]). We define this class of functions as self-concordant like functions.

Definition 1.20 *(Multivariate Self-Concordant Like Function). Let $M_f \geq 0$ and $\mathcal{D} \subseteq \mathbb{R}^d$. The function $f : \mathcal{D} \to \mathbb{R}$ is called self-concordant like if and only if $\varphi(\theta) := f(\mathbf{x} + \theta\mathbf{d})$ is self-concordant like at $\theta = 0$, for all $\mathbf{x} \in \mathcal{D}$ and $\mathbf{d} \in \mathbb{R}^d$*

$$\left|\varphi'''(0)\right| \;\leq\; M_f\,\|\mathbf{d}\|_2\,\varphi''(0) \quad \forall \mathbf{x} \in \mathcal{D} \,,\;\; \forall \mathbf{d} \in \mathbb{R}^d \,. \tag{1.31}$$

In particular, given $f(\mathbf{x}) = \sum_{j=1}^{d} \log(1 + \exp(-x_j))$, we have that

$$\left|\varphi'''(0)\right| \;\leq\; \|\mathbf{d}\|_2 \, \varphi''(0)\,. \tag{1.32}$$

Proposition 1.21 *(Taylor Expansion for Self-Concordant Like Functions). Let $f : \mathcal{D} \to \mathbb{R}$, with $\mathcal{D} \subseteq \mathbb{R}^d$, be a self-concordant like function with constant $M_f \geq 0$. For all $\mathbf{x}$, $\mathbf{d} \in \mathcal{D}$, we have*

$$f(\mathbf{x}+\mathbf{d}) \leq f(\mathbf{x}) + \mathbf{d}^\top \nabla f(\mathbf{x}) + \frac{\mathbf{d}^\top \nabla^2 f(\mathbf{x}) \mathbf{d}}{M_f^2 \|\mathbf{d}\|_2^2} \left(\exp(M_f \|\mathbf{d}\|_2) - M_f \|\mathbf{d}\|_2 - 1\right),$$

$$\nabla^2 f(\mathbf{x}+\mathbf{d}) \leq \exp(M_f \|\mathbf{d}\|_2) \nabla^2 f(\mathbf{x})\,.$$

Proof For the proof, we refer to Bach's paper [1]. □

Proposition 1.22 *(Second Order Taylor Equality). Let $f : \mathcal{D} \to \mathbb{R}$, with $\mathcal{D} \subseteq \mathbb{R}^d$, be a twice differentiable function. Then for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$, there exist $\theta \in [0,1]$ such that for $\mathbf{z} = (1-\theta)\mathbf{x} + \theta\mathbf{y}$ we have*

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y}-\mathbf{x}) + \frac{1}{2}(\mathbf{y}-\mathbf{x})^\top \nabla^2 f(\mathbf{z})(\mathbf{y}-\mathbf{x})\,.$$

Chapter 2

CoCoA

2.1 Setting

The CoCoA framework, as described in [13], has been designed to address the specific setting of regularized ERM problems. Already with [2], the already existing CoCoA+ framework, described in [7], was extended to problems with non-strongly convex regularizers. Consequently, after this generalization, CoCoA can address a much broader class of optimization problems, including also Lasso and ℓ_1-logistic regression, and it allows for the flexibility of distributing the data by either features or data points. Its communication efficiency and ability to reuse arbitrarily accurate local solvers, make this framework one of the most promising to address the need of analysing and processing big data sets.

2.1.1 Primal-Dual Setting

As already mentioned, CoCoA addresses problems with a specific structure, e.g. (1.1). In the literature, many different methods have been analyzed to solve regularized ERM problems. Without going too much into details, they can be divided in two major categories: primal methods, which run directly on the primal formulation, and dual methods, as the famous stochastic dual coordinate ascent methods [12], which optimize the dual problem instead. CoCoA acts as both a primal and a dual method since, depending on the problem's specific properties, it addresses either the primal or the dual formulation, or, in some cases, it could potentially address both of them, letting free choice to the user. Indeed, by employing the Fenchel-Rockafellar duality, the CoCoA framework is based on the following abstraction: considering the properties of the functions involved, the problem (1.1) is mapped to one of the following formulations:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad [\mathcal{O}_A(\boldsymbol{\alpha}) := f(A\boldsymbol{\alpha}) + g(\boldsymbol{\alpha})] \,, \tag{2.1}$$

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \left[\mathcal{O}_B(\mathbf{w}) := g^*(-A^\top \mathbf{w}) + f^*(\mathbf{w})\right] \,, \tag{2.2}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\mathbf{w} \in \mathbb{R}^d$ are parameter vectors, and $A := [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ is a data matrix with column vectors $\mathbf{x}_i \in \mathbb{R}^d$, $i \in [n]$. In addition, the problems (2.1) and (2.2) are Fenchel-Rockafellar dual problems (see Section 1.3.2 for more details). Once the problem has been mapped, CoCoA acts by solving (2.1): therefore, if the original ERM was mapped to (2.2), it is necessary to derive its Fenchel-Rockafellar dual formulation. Consequently, CoCoA acts as a primal method when the problem solved is mapped directly to (2.1), and as a dual one in the other case. In addition, as underlined in the paper [2], the primal-dual relationship between (2.1) and (2.2) provides the chance to compute the duality gap

$$G(\boldsymbol{\alpha}) := \mathcal{O}_A(\boldsymbol{\alpha}) - [-\mathcal{O}_B(\mathbf{w})] \,. \tag{2.3}$$

From the definition of dual problem, the following inequality always holds

$$G(\boldsymbol{\alpha}) \geq 0 \,, \tag{2.4}$$

where the equality is achieved at the optimum. The possibility of computing the duality gap is really useful in practice, since, with its property of being an upper bound on the suboptimality of the current solution, it can be used both as a stopping condition and as a certificate of the approximation quality.

2.1.2 Assumptions

The underlying assumptions required to accomodate an input problem in the CoCoA framework are the following: regarding the mapping to problem (2.1), f is assumed to be $(\frac{1}{\tau})$-smooth and convex, and $g(\boldsymbol{\alpha}) = \sum_{i=1}^n g_i(\alpha_i)$ convex and separable; for the mapping to problem (2.2), f^* is supposed to be τ-strongly convex and $g^*(-A^\top \boldsymbol{\alpha}) = \sum_{i=1}^n g_i^*(-\mathbf{x}_i^\top \mathbf{w})$ convex and separable.

Examples

In the following we will show how some of the major examples for regularized ERM can be mapped to the CoCoA framework by using (2.1) and (2.2). Regarding the notation: $A := [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ is the data matrix, and $\mathbf{b}$ is a vector of size equal to the number of data points, therefore either d or n depending on the CoCoA mapping. Indeed, the role of n and d depends

on the way the original problem is mapped to one of the two dual CoCoA problem. In particular, for primal-CoCoA, n is the number of features and d the number of data points; while, in the dual setting, n is the number of data points and d the number of features.

1. **Lasso**
   Starting from the general Lasso formulation (1.2), since the ℓ_1-norm is non-strongly convex, it is only possible a mapping to (2.1), where $f(A\boldsymbol{\alpha}) = \frac{1}{2}\|A\boldsymbol{\alpha} - \mathbf{b}\|^2$ and $g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} g_i(\alpha_i) = \sum_{i=1}^{n} \lambda|\alpha_i|$. In this case, CoCoA acts as a primal method, consequently, $\mathbf{b} \in \mathbb{R}^d$.

2. **Elastic Net Regression**
   Given the assumptions fulfilled by elastic net regularized least squares regression (1.3), the problem can be arbitrarily mapped to either (2.1) or (2.2). In the first case, $f(A\boldsymbol{\alpha}) = \frac{1}{2}\|A\boldsymbol{\alpha} - \mathbf{b}\|_2^2$ and $g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} g_i(\alpha_i) = \sum_{i=1}^{n} \lambda(\frac{\eta}{2}|\alpha_i|^2 + (1-\eta)|\alpha_i|)$, where $\mathbf{b} \in \mathbb{R}^d$, and n and d are the number of features and data points, respectively; in the second case, $g(-A^\top \mathbf{w}) = \sum_{i=1}^{n} g_i^*(-\mathbf{x}_i^\top \mathbf{w}) = \sum_{i=1}^{n} \frac{1}{2}(\mathbf{x}_i^\top \mathbf{w} - b_i)^2$ and $f^*(\mathbf{w}) = \lambda(\frac{\eta}{2}\|\mathbf{w}\|_2^2 + (1-\eta)\|\mathbf{w}\|_1)$, where $\mathbf{b} \in \mathbb{R}^n$, since now d is the number of features and n the number of data points. Consequently, CoCoA can potentially acts both as a primal or a dual method.

3. **ℓ_2-Logistic Regression**
   Regarding the case of ℓ_2-regularized logisitc regression (1.4), as for the elastic net regression, CoCoA can potentially acts as either a primal or a dual method, since both the mappings are possible. In the case of a mapping to (2.1), $f(A\boldsymbol{\alpha}) = \sum_{j=1}^{d} \log(1 + \exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha}))$ and $g(\boldsymbol{\alpha}) = \frac{\lambda}{2}\|\boldsymbol{\alpha}\|_2^2$, where d is the number of data points and n the number of features; in the second case, $g^*(-A^\top \mathbf{w}) = \sum_{i=1}^{n} \log(1 + \exp(-b_i \mathbf{x}_i^\top \mathbf{w}))$ and $f^*(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$, where the role of n and d is switched.

4. **ℓ_1-Logistic Regression**
   Since ℓ_1-norm is non-strongly convex, the ony possible mapping for (1.5) is to (2.1). Therefore, CoCoA always acts as a primal method, by addressing the problem formulation in its primal form. In particular, $f(A\boldsymbol{\alpha}) = \sum_{j=1}^{d} \log(1 + \exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha}))$ and $g(\boldsymbol{\alpha}) = \lambda\|\boldsymbol{\alpha}\|_1$, where d is the number of data points and n the number of features.

5. **ℓ_2-Support Vector Machine**
   Regarding the smoothly regularized SVM case (1.6), the loss function is now clearly non-smooth, therefore the problem can not be adjusted in the CoCoA primal framework, but, by exploiting the strong convexity of the ℓ_2-norm regularizer, it is possible to map the problem to (2.2). In particular, $g^*(-A^\top \mathbf{w}) = \frac{1}{d}\sum_{j=1}^{d} \max\{0, 1 - b_j \mathbf{y}_j^\top \mathbf{u}\}$ and $f^*(\mathbf{w}) = \lambda\|\mathbf{w}\|_2^2$, where d is the number of features, and n the number of data points.

6. **ℓ_1-Support Vector Machine**
   SVM with a ℓ_1-norm regularizer can not be solved in the CoCoA framework, since the loss function is non smooth and, differently from the case with the ℓ_2-norm, this time it is not possible to exploit the strong convexity of the regularizer: the presence of the ℓ_1-norm makes possible to adjust (1.7) to neither (2.1) nor (2.2).

2.2 Algorithmic Framework

CoCoA is very versatile in its way of dealing with big size regularized ERM problems: indeed, it can be used as either a parallel framework or a distributed one, depending on a foundamental assumption regarding the data matrix A. If the data matrix is too big to be stored in a single machine, then it is necessary to operate in a distributed framework, where A is partitioned across different machines. On the other hand, in the case that A is stored in a single shared memory, but it is still too big in size to be efficiently operated on a single processor, then CoCoA can be used as a parallel framework, where multiple processors work in parallel, but they all have access to the same shared memory. In this thesis, we will first focus on its distributed configuration, and then, in Chapter 5, we will analyze the parallel framework.

2.2.1 Data Partitioning

The data is distributed across machines according to column-wise partitions of the original matrix $A \in \mathbb{R}^{d \times n}$. Therefore, data are distributed either by features or data points, depending on the way the problem is mapped into CoCoA framework.

In the distributed environment, the data matrix A is physically stored in K different machines according to a partition $\{\mathcal{P}_k\}_{k=1}^{K}$ of the columns of $A \in \mathbf{R}^{d \times n}$. Given $n_k = |\mathcal{P}_k|$ to be the size of the k partition, for each of the K machines we define $\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^n$ as the local vector with elements $(\boldsymbol{\alpha}_{[k]})_i := \alpha_i$ $\forall i \in \mathcal{P}_k$, and $(\boldsymbol{\alpha}_{[k]})_i := 0$ otherwise. Analogously, $A_{[k]} \in \mathbb{R}^{d \times n}$ is the matrix of locally available data, with elements $(A_{[k]})_{[i,j]} := A_{[i,j]}$ $\forall i \in [d]$ and $\forall j \in \mathcal{P}_k$, and $(A_{[k]})_{[i,j]} := A_{[i,j]}$ $\forall i \in [d]$ and $\forall j \notin \mathcal{P}_k$.

2.2.2 Data-Local Subproblems

Once the regularized ERM problem (1.1) has been mapped to either (2.1) or (2.2), CoCoA is addressing (2.1). Now a question arises regarding the way CoCoA is distributing the computation among different machines. Indeed, problem (2.1) is not separable. Therefore, instead of addressing directly (2.1), in each iteration CoCoA optimizes a separable local approximation of it.

Notation

A small notation note, before proceeding further with the analysis: let $v : \mathbb{R}^n \to \mathbb{R}^d$ be the linear map associated with the data matrix $A \in \mathbb{R}^{d \times n}$, and $\ell : \mathbb{R}^d \to \mathbb{R}$ be the composition $\ell = f \circ v$. As it has been shown in (1.17), given that by assumption f is $(\frac{1}{\tau})$-smooth and v is a linear map, ℓ is smooth and its constant of smoothness is directly proportional to $(\frac{1}{\tau})$.
Similarly to the proximal gradient method approach (see section 1.2.3), the local approximation built by CoCoA in each iteration is given by a quadratic local approximation of the smooth component ℓ, plus the original g:

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \gamma \underset{\Delta\boldsymbol{\alpha} \in \mathbb{R}^n}{\arg\min} \left\{ f(\mathbf{v}^{(t)}) + \nabla f(\mathbf{v}^{(t)})^\top A \Delta\boldsymbol{\alpha} + \frac{1}{2\tau} \Delta\boldsymbol{\alpha}^\top A^\top I A \Delta\boldsymbol{\alpha} + g(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \right\}, \tag{2.5}$$

where $\gamma \in (0, 1]$ is a certain step length.

The problem (2.5) is clearly not separable yet. Consequently, another approximation is done to operate on it in a distributed fashion: the full data matrix $A^\top A$ is replaced with a block diagonal approximation of it, where the blocks are determined by the way the data matrix A is partitioned among the machines, in order to avoid extra communication costs. A parameter σ'_C is introduced to account for the loss of information: as it will be shown in more details in Section 2.3, its role is foundamental for the convergence of the method, as well as its performances. The resulting problem is

$$\begin{aligned} \boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \gamma \underset{\Delta\boldsymbol{\alpha} \in \mathbb{R}^n}{\arg\min} \Big\{ & f(\mathbf{v}^{(t)}) + \nabla f(\mathbf{v}^{(t)})^\top A \Delta\boldsymbol{\alpha} + \frac{\sigma'_C}{2\tau} \Delta\boldsymbol{\alpha}^\top \begin{bmatrix} A_{[1]}^\top A_{[1]} & & 0 \\ & \ddots & \\ 0 & & A_{[K]}^\top A_{[K]} \end{bmatrix} \Delta\boldsymbol{\alpha} \\ & + g(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \Big\}. \end{aligned} \tag{2.6}$$

The problem approximation (2.6) is now separable across the K machines. At iteration $(t+1)$, the machine k, that stores the submatrix $A_{[k]} \in \mathbb{R}^{d \times n_k}$, addresses the following local subproblem:

$$\min_{\Delta\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^{n_k}} \mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}) = \nabla f(\mathbf{v})^\top A_{[k]} \Delta\boldsymbol{\alpha}_{[k]} + \frac{\sigma'_C}{2\tau} \| A_{[k]} \Delta\boldsymbol{\alpha}_{[k]} \|_2^2 + \sum_{i=1}^{n_k} g_i(\alpha_i + (\Delta\boldsymbol{\alpha}_{[k]})_i). \tag{2.7}$$

Local subproblems depends only on the locally available data submatrices $A_{[k]}$ $\forall k \in [K]$, and the vector of shared global information regarding the previous iteration $\mathbf{v}^{(t)} = A\boldsymbol{\alpha}^{(t)}$.

Algorithm 1 Distributed-CoCoA

Input: data matrix A distributed column-wise across K machines, aggregation parameter $\gamma \in (0, 1]$, and parameter σ'_C for the local subproblems $\mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v})$
central machine initialization $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{v}^{(0)} := \mathbf{0} \in \mathbb{R}^d$
set $t = -1$
while not converged **do**
  $t = t + 1$
  for $k \in \{1, 2, \ldots, K\}$ **in parallel over machines do**
    compute a Θ-approximate solution $\Delta\boldsymbol{\alpha}_{[k]}$ of the local subproblem $\mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}^{(t)}) = (2.7)$
    update the local variable $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \gamma\Delta\boldsymbol{\alpha}_{[k]}$
    communicate to the central machine $\mathbf{v}_k = A_{[k]}\boldsymbol{\alpha}_{[k]}$
  end for
  central machine updates the global information $\mathbf{v}^{(t+1)} := \mathbf{v}^{(t)} + \gamma \sum_{k=1}^{K} \mathbf{v}_k$
end while

2.2.3 Reusability of Single Machines Solvers

One of the strongest points of the CoCoA framework is that it does not require the subproblems to be solved exactly but it allows the use of arbitrarily accurate local solvers instead.

Proposition 2.1 *(Θ-approximate solution). We assume that there exists $\Theta \in [0, 1)$ such that $\forall k \in [K]$, the local solver at any outer iteration t produces a possibly randomized approximate solution $\Delta\boldsymbol{\alpha}_{[k]}$, which satisfies*

$$\mathbb{E}\left[\mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}) - \mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}^{\star}_{[k]}; A_{[k]}, \mathbf{v})\right] \leq \Theta\left(\mathcal{G}_k^{\sigma'_C}(\mathbf{0}; A_{[k]}, \mathbf{v}) - \mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}^{\star}_{[k]}; A_{[k]}, \mathbf{v})\right),$$

where

$$\Delta\boldsymbol{\alpha}^{\star}_{[k]} \in \arg\min_{\Delta\boldsymbol{\alpha} \in \mathbb{R}^n} \mathcal{G}_k^{\sigma'_C}\left(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}\right).$$

Remark 2.2 *In the distributed environment, since communications are expensive, the time spent in solving the local subproblems should be a trade off between the required time of a communication round and computational costs.*

2.2.4 Bounded Support Modification

In order to fulfill all the assumptions required for the validity of the convergence results exposed in the next section, in addition to the starting requirements (see 2.1.2), g_i^* should also be L-Lipschitz, $\forall i \in [n]$. Given the

dual relation between Lipschitzness and L-bounded, this last requirement is equivalent to the assumption of g_i having an L-bounded support, $\forall i \in [n]$. Consequently, each one of the g_i functions is replaced by the following modified function:

$$\bar{g}_i(\alpha) := \begin{cases} g_i(\cdot) & : \ \alpha \in [-L, L] \\ +\infty & : \ \text{otherwise}\,. \end{cases} \tag{2.8}$$

For L large enough, this change is not affecting the problem formulation. Hence, it preserves all the original solutions, since (2.8) corresponds to a set of weak constraints that are always inactive in the region of interest. This is also the main difference with respect to the generally applied smoothing techniques, see Nesterov's paper [8], that, contrarily, are changing the problem formulation actively and, therefore, its solutions. In addition, it is generally very difficult to tune the smoothing component. In conclusion, this "Lipschitzing" technique allows us for the applicability of the convergence reults that are presented in the next chapter to general convex g functions, without actively modifying the problem framework.

Safe Choice of L

In particular, consider the problem

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad f(A\boldsymbol{\alpha}) + \lambda \|\boldsymbol{\alpha}\|\,,$$

where $f(A\boldsymbol{\alpha})$ is a convex, smooth and non-negative loss function, and $\|\cdot\|$ is a generic norm. Then, for any given monotone algorithm initialized to $\mathbf{0} \in \mathbb{R}^n$, we have that

$$\|\boldsymbol{\alpha}\| \leq \frac{1}{\lambda} f(\mathbf{0})\,.$$

This expression can be easily used to tune the L parameter such that the introduced contraints are never active in the region where our algorithm will work.

2.3 Convergence Analysis

This section is opened by a formal definition for the parameter σ'_C, introduced in (2.6), whose role in the convergence of the method is foundamental. Indeed, σ'_C can not only affect the convergence rate substantially, but also determine whether or not CoCoA will achive convergence. Then, we will formally analyze the convergence rate of CoCoA. The proof is based on two foundamental lemmas, which then will lead to the final convergence

theorem. In particular, the first lemma relates the progress of the local subproblems to the original objective function $\mathcal{O}_A(\cdot)$, and the second one quantifies the effect of one iteration of CoCoA on the duality gap for any chosen local solver with approximation Θ. Finally, in the theorem, an upper bound on the number of outer iterations is derived.

2.3.1 σ'_C Parameter

A rigurous definition for the σ'_C parameter is provided.

Definition 2.3 *Given a step size $\gamma \in (0, 1]$, the value of the σ'_C parameter should be chosen accordingly to the following inequlity*

$$\sigma'_C \geq (\sigma'_C)_{min} := \gamma \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{\|A\Delta\boldsymbol{\alpha}\|^2}{\sum_{k=1}^K \|A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}\|^2} \,. \tag{2.9}$$

It is also possible to define a data-independent safe bound: indeed, by choosing $\sigma'_C \geq \sigma' := \gamma K$, the inequality (2.9) *always holds.*

The reasons behind this choice comes from lemma (2.4). In a few words, σ'_C should ensure that the local quadratic approximation always lies above the one obtained by considering the real Hessian matrix of ℓ. In so doing, convergence is preserved.

2.3.2 Lemmas and Theorem of Convergence

Note: The results of the following lemmas and theorem are valid for any choice of σ'_C such that equation (2.9) holds. In addition, all the working assumptions of CoCoA framework regarding f and g are required to be fulfilled.

Lemma 2.4 *For any $\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha} \in \mathbb{R}^n, \mathbf{v} = \mathbf{v}(\boldsymbol{\alpha}) := A\boldsymbol{\alpha}$, $\gamma \in (0, 1]$, it holds that*

$$\mathcal{O}_A(\boldsymbol{\alpha} + \gamma \sum_{k=1}^K \Delta\boldsymbol{\alpha}_{[k]}) \leq (1-\gamma)\mathcal{O}_A(\boldsymbol{\alpha}) + \gamma \sum_{k=1}^K \mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}) \,. \tag{2.10}$$

Proof In this proof, the assumption of $(\frac{1}{\tau})$-smoothness of f is foundamental. After one outer iteration, CoCoA performs on the objective $\mathcal{O}_A(\cdot)$ the following update:

$$\mathcal{O}_A(\boldsymbol{\alpha} + \gamma \sum_{k=1}^K \Delta\boldsymbol{\alpha}_{[k]}) = \underbrace{f(\mathbf{v}(\boldsymbol{\alpha} + \gamma \sum_{k=1}^K \Delta\boldsymbol{\alpha}_{[k]}))}_{\text{A}} + \underbrace{\sum_{i=1}^n g_i(\alpha_i + \gamma(\sum_{k=1}^K \Delta\boldsymbol{\alpha}_{[k]})_i)}_{\text{B}} \,. \tag{2.11}$$

We proceed by bounding the terms A and B separatly.

A term: The bound on A is found by exploiting the $(\frac{1}{\tau})$-smoothness property of f:

$$\begin{aligned}
A &= f(\mathbf{v}(\boldsymbol{\alpha} + \gamma \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]})) = f(\mathbf{v}(\boldsymbol{\alpha}) + \gamma \sum_{k=1}^{K} \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]})) \\
&\overset{(\frac{1}{\tau})-\text{smoothness}}{\leq} f(\mathbf{v}(\boldsymbol{\alpha})) + \gamma \nabla f(\mathbf{v}(\boldsymbol{\alpha}))^\top \mathbf{v}(\Delta\boldsymbol{\alpha}) + \frac{\gamma^2}{2\tau} \mathbf{v}(\Delta\boldsymbol{\alpha})^\top I \mathbf{v}(\Delta\boldsymbol{\alpha}) \\
&\overset{\mathbf{w}:=\nabla f(A\boldsymbol{\alpha})}{=} f(\mathbf{v}(\boldsymbol{\alpha})) + \sum_{k=1}^{K} \gamma \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]})^\top \mathbf{w}(\boldsymbol{\alpha}) + \frac{\gamma^2}{2\tau} \| \sum_{k=1}^{K} \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]}) \|^2 \\
&\overset{\text{safe value } \sigma'_C}{\leq} f(\mathbf{v}(\boldsymbol{\alpha})) + \sum_{k=1}^{K} \gamma \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]})^\top \mathbf{w}(\boldsymbol{\alpha}) + \frac{\gamma \sigma'_C}{2\tau} \sum_{k=1}^{K} \| \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]}) \|^2 .
\end{aligned}$$

B term: To bound the B term, we use the Jensen's inequality:

$$\begin{aligned}
B = \sum_{k=1}^{K} \left(\sum_{i \in \mathcal{P}_k} g_i(\alpha_i + \gamma(\nabla\boldsymbol{\alpha}_{[k]})_i) \right) &= \sum_{k=1}^{K} \left(\sum_{i \in \mathcal{P}_k} g_i((1-\gamma)\alpha_i + \gamma(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}_{[k]})_i) \right) \\
&\leq (1-\gamma) g(\boldsymbol{\alpha}) + \gamma g(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) .
\end{aligned}$$

Now, we simply plug the bound on A and B back into equation (2.11), and add and subtract the term $\gamma f(\mathbf{v}(\boldsymbol{\alpha}))$:

$$\mathcal{O}_A(\boldsymbol{\alpha} + \gamma \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]}) \leq f(\mathbf{v}(\boldsymbol{\alpha})) \pm \gamma f(\mathbf{v}(\boldsymbol{\alpha})) + \sum_{k=1}^{K} \gamma \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]})^\top \mathbf{w}(\boldsymbol{\alpha}) + \frac{\gamma \sigma'_C}{2\tau} \sum_{k=1}^{K} \| \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]}) \|^2$$

$$\begin{aligned}
&+ \sum_{k=1}^{K} \sum_{i \in \mathcal{P}_k} \Big((1-\gamma) g_i(\alpha_i) + \gamma g_i(\boldsymbol{\alpha}_i + \Delta\boldsymbol{\alpha}_{[k]_i}) \Big) \\
&= \underbrace{(1-\gamma) f(\mathbf{v}(\boldsymbol{\alpha})) + \sum_{k=1}^{K} \left(\sum_{i \in \mathcal{P}_k} (1-\gamma) g_i(\alpha_i) \right)}_{(1-\gamma)\mathcal{O}_A(\boldsymbol{\alpha})} \\
&+ \gamma \sum_{k=1}^{K} \left(\frac{1}{K} f(\mathbf{v}(\boldsymbol{\alpha})) + \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]})^\top \mathbf{w}(\boldsymbol{\alpha}) + \frac{\sigma'_C}{2\tau} \| \mathbf{v}(\Delta\boldsymbol{\alpha})_{[k]}) \|^2 + \sum_{i \in \mathcal{P}_k} g_i(\boldsymbol{\alpha}_i + \Delta\boldsymbol{\alpha}_{[k]_i}) \right) \\
&= (1-\gamma) \mathcal{O}_A(\boldsymbol{\alpha}) + \gamma \sum_{k=1}^{K} \mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}) ,
\end{aligned}$$

where the last equality is based on the definition of the local subproblems in (2.7). □

Lemma 2.5 *Let g_i be convex and σ'_C be the safe data independent value. Then, given proposition 2.1 for the local solvers, and any $s \in [0,1]$, for all iterations of CoCoA it holds that*

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] \geq \gamma(1-\Theta)\left(sG(\boldsymbol{\alpha}^{(t)}) - \frac{\sigma'_C s^2}{2\tau}R^{(t)}\right), \tag{2.12}$$

where

$$R^{(t)} := \sum_{k=1}^{K} \|A_{[k]}(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})_{[k]}\|^2, \tag{2.13}$$

for $\mathbf{u}^{(t)} \in \mathbb{R}^n$ with

$$u_i^{(t)} \in \partial g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}^{(t)})). \tag{2.14}$$

Proof For simplicity, we will drop the apex t referring to the iteration, unless really necessary to avoid wrong interpretations.
By applying lemma (2.4), we obtain that:

$$\begin{aligned}\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] &= \mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}) - \mathcal{O}_A(\boldsymbol{\alpha} + \gamma\sum_{k=1}^{K}\Delta\boldsymbol{\alpha}_{[k]})\right] \\ &\geq \gamma(1-\Theta)\left(\underbrace{\mathcal{O}_A(\boldsymbol{\alpha}) - \sum_{k=1}^{K}\mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}^\star_{[k]}; A_{[k]}, v)}_{\mathrm{C}}\right). \end{aligned}\tag{2.15}$$

Now, we look for a bound on C by applying Jensen's inequality to the functions g_i, and by substituting $(\Delta\boldsymbol{\alpha}_{[k]})_i^\star$ with the subgradient $s(u_i - \alpha_i)$, where u_i satifies the optimality conditions by assumption, and where $\Delta\boldsymbol{\alpha}^\star = \sum_{k=1}^{K}\Delta\boldsymbol{\alpha}_{[k]}$. This gives us:

$$\begin{aligned} C &= \sum_{i=1}^{n}(g_i(\alpha_i) - g_i(\alpha_i + \Delta\boldsymbol{\alpha}_i^\star)) - (A\Delta\boldsymbol{\alpha}^\star)^\top\mathbf{w}(\boldsymbol{\alpha}) - \sum_{k=1}^{K}\frac{\sigma'_C}{2\tau}\|A_{[k]}\Delta\boldsymbol{\alpha}^\star_{[k]}\|^2 \\ &\geq \sum_{i=1}^{n}(sg_i(\alpha_i) - sg_i(u_i)) - A(s(\mathbf{u}-\boldsymbol{\alpha}))^\top\mathbf{w}(\boldsymbol{\alpha}) - \sum_{k=1}^{K}\frac{\sigma'_C}{2\tau}\|A_{[k]}(s(\mathbf{u}-\boldsymbol{\alpha})_{[k]})\|^2. \end{aligned}\tag{2.16}$$

We next analyze the duality gap, and by plugging in the definition of problem (2.1) and problem (2.2), and by exploiting the definition of convex conjugate functions (1.1), we obtain:

$$G(\boldsymbol{\alpha}) := \mathcal{O}_A(\boldsymbol{\alpha}) - \mathcal{O}_B(\mathbf{w}(\boldsymbol{\alpha})) = \sum_{i=1}^{n} \left(g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) + g_i(\alpha_i) \right) + f^*(\nabla f(A\boldsymbol{\alpha})) + f(A\boldsymbol{\alpha})$$
$$= \sum_{i=1}^{n} \left(g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) + g_i(\alpha_i) + \alpha_i \mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}) \right) .$$

From the maximal property of convex conjugate functions, we have that:

$$g_i(u_i) = u_i(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) - g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) , \tag{2.17}$$

and, consequently, by plugging (2.17) back into (2.16) we obtain:

$$C \geq \sum_{i=1}^{n} \left(s g_i(\alpha_i) - s u_i(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) + s g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) \right)$$
$$- A(s(\mathbf{u} - \boldsymbol{\alpha}))^\top \mathbf{w}(\boldsymbol{\alpha}) - \sum_{k=1}^{K} \frac{\sigma_C'}{2\tau} \| A_{[k]}(s(\mathbf{u} - \boldsymbol{\alpha})_{[k]}) \|^2$$
$$= sG(\boldsymbol{\alpha}) - \frac{\sigma_C' s^2}{2\tau} \sum_{k=1}^{K} \| A_{[k]}(\mathbf{u} - \boldsymbol{\alpha})_{[k]} \| . \tag{2.18}$$

□

The conclusion follows straightforwad, by plugging (2.18) back into (2.15).

The following lemma is proving that the $R^{(t)}$ term is bounded above, assuming g_i^* being L-Lipschitz continuous $\forall i \in [n]$, or, equivalently, that g_i have an L-bounded support $\forall i \in [n]$.

Lemma 2.6 *If g_i^* are L-Lipschitz continuous for all $i \in [n]$, then*

$$R^{(t)} \leq 4L^2 \underbrace{\sum_{k=1}^{K} \sigma_k n_k}_{=:\sigma} \quad \forall t , \tag{2.19}$$

where σ_k is nothing but the induced norm of $A_{[k]}$

$$\sigma_k := \max_{\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^n} \frac{\| A_{[k]} \boldsymbol{\alpha}_{[k]} \|^2}{\| \boldsymbol{\alpha}_{[k]} \|^2} \quad \forall k \in [K] . \tag{2.20}$$

Proof The proof is mainly based on the fact that, by assuming g_i^* to be L-Lipschitz continuous, we are also assuming g_i to have an L-bounded support,

since, given that g_i and g_i^* are a convex conjugate pair, these requirements are equivalent.

$$R^{(t)} = \sum_{k=1}^{K} \|A_{[k]}(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})_{[k]}\|^2 \leq \sum_{k=1}^{K} \|A_{[k]}\|^2 \|(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})\|^2$$
$$= \sum_{k=1}^{K} \sigma_k \|(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})_{[k]}\|^2 \leq 4L^2 \sum_{k=1}^{K} \sigma_k n_k \,. \qquad \square$$

The following theorem is using the results from the previous lemmas to demonstrate that CoCoA is achieving convergence to the optimal solution of problem (2.1) by solving with an arbitrary accuracy the local subproblems (2.7).

Theorem 2.7 *Consider Algorithm* (1) *and a local solver of quality* Θ*. Let* $g_i^*(\cdot)$ *be L-Lipschitz continuous, and* $\varepsilon_G \geq 0$ *be the desired duality gap value. Then, after T iterations, where*

$$T \geq T_0 + \max\left\{ \left\lceil \frac{1}{\gamma(1-\Theta)} \right\rceil, \left\lceil \frac{4L^2\sigma\sigma'_C}{\tau\varepsilon_G\gamma(1-\Theta)} \right\rceil \right\} \tag{2.21}$$

$$T_0 \geq t_0 + \left\lceil \left[\frac{2}{\gamma(1-\Theta)} \left(\frac{8L^2\sigma\sigma'_C}{\tau\varepsilon_G} - 1 \right) \right]^+ \right\rceil \tag{2.22}$$

$$t_0 \geq \max\left(0 \;,\; \left\lceil \frac{1}{\gamma(1-\Theta)} \log\left(\frac{\tau(\mathcal{O}_A(\boldsymbol{\alpha}^{(0)}) - \mathcal{O}_A(\boldsymbol{\alpha}^*))}{2L^2\sigma\sigma'_C} \right) \right\rceil \right) \tag{2.23}$$

we have that the expected duality gap satifies

$$\mathbb{E}\left[\mathcal{O}_A(\bar{\boldsymbol{\alpha}}) - (-\mathcal{O}_B(\mathbf{w}(\bar{\boldsymbol{\alpha}})))\right] \leq \varepsilon_G \tag{2.24}$$

where $\bar{\boldsymbol{\alpha}}$ *is the averaged iterate*

$$\bar{\boldsymbol{\alpha}} := \frac{1}{T - T_0} \sum_{t=T_0}^{T-1} \boldsymbol{\alpha}^{(t)} \,. \tag{2.25}$$

Proof We begin by bounding above the expected suboptimality at iteration $t+1$. In so doing, we apply lemma (2.5) and the weak duality theorem

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] = \mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t)})\right] + \mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)$$
$$\leq -\gamma(1-\Theta)\left(sG(\boldsymbol{\alpha}^{(t)}) - \frac{\sigma'_C s^2}{2\tau} 4L^2\sigma \right) + \mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)$$
$$\leq (1 - \gamma(1-\Theta)s)\left(\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star) \right) + \gamma(1-\Theta)\frac{\sigma'_C s^2}{2\tau} 4L^2\sigma \,. \tag{2.26}$$

By recursively applying (2.26), we obtain

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] \leq (1-\gamma(1-\Theta)s)^t(\mathcal{O}_A(\boldsymbol{\alpha}^{(0)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)) + s\frac{4L^2\sigma\sigma'_C}{2\tau}\,.$$

Then, given $s = 1$ and $t = t_0 := \max\left\{0\ ,\ \left\lceil \frac{1}{\gamma(1-\Theta)}\log(2(\mathcal{O}_A(\boldsymbol{\alpha}^{(0)})) - \mathcal{O}_A(\boldsymbol{\alpha}^\star))/4L^2\sigma\sigma'_C \right\rceil\right\}$, we want to inductively show that the following bound holds

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^t) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] \leq \frac{4L^2\sigma\sigma'_C}{\tau(1+\frac{1}{2}\gamma(1-\Theta)(t-t_0))} \quad \forall t \geq t_0\,. \tag{2.27}$$

Clearly, for the base case, (2.27) holds

$$\begin{aligned}\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] &\leq (1-\gamma(1-\Theta))^{t_0}(\mathcal{O}_A(\boldsymbol{\alpha}^{(0)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)) + \frac{4L^2\sigma\sigma'_C}{2\tau} \\ &\leq \frac{4L^2\sigma\sigma'_C}{\tau}\,.\end{aligned}$$

Assuming that (2.27) holds $\forall t \geq t_0$, we want to prove that it holds also for $t+1$. By considering

$$s = \frac{1}{1+\frac{1}{2}\gamma(1-\Theta)(t-t_0)} \quad \in [0,1]\,, \tag{2.28}$$

and by applying the bounds (2.26) and (2.27), we obtain

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)} - \mathcal{O}_A(\boldsymbol{\alpha}^\star))\right] \leq \frac{4L^2\sigma\sigma'_C}{\tau}\underbrace{\left(\frac{1+\frac{1}{2}\gamma(1-\Theta)(t-t_0) - \frac{1}{2}\gamma(1-\Theta)}{(1+\frac{1}{2}\gamma(1-\Theta)(t-t_0))^2}\right)}_{\mathrm{D}}\,.$$

We then upperbound the term D by using the fact that the geometric mean is less or equal than the arithmetic mean

$$\begin{aligned}D &= \frac{1}{1+\frac{1}{2}\gamma(1-\Theta)(t+1-t_0)}\frac{(1+\frac{1}{2}\gamma(1-\Theta)(t+1-t_0))(1+\frac{1}{2}\gamma(1-\Theta)(t-1-t_0))}{(1+\frac{1}{2}\gamma(1-\Theta)(t-t_0))^2} \\ &\leq \frac{1}{1+\frac{1}{2}\gamma(1-\Theta)(t+1-t_0)}\end{aligned}$$

Given $\bar{\boldsymbol{\alpha}}$ defined as in (2.25), by using the results of lemma (2.4) and lemma (2.5), we obtain

$$\mathbb{E}\left[G(\bar{\boldsymbol{\alpha}})\right] = \mathbb{E}\left[G\left(\sum_{t=T_0}^{T-1}\frac{1}{T-T_0}\boldsymbol{\alpha}^{(t)}\right)\right] \leq \frac{1}{T-T_0}\mathbb{E}\left[\sum_{t=T_0}^{T-1}G(\boldsymbol{\alpha}^{(t)})\right]$$
$$\leq \frac{1}{\gamma(1-\Theta)s}\frac{1}{T-T_0}\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(T_0)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] + \frac{4L^2\sigma\sigma'_C s}{2\tau}\,.$$

If $T \geq \left\lceil \frac{1}{\gamma(1-\Theta)} \right\rceil + T_0$ such that $T_0 \geq t_0$, we have

$$\mathbb{E}\left[G(\bar{\boldsymbol{\alpha}})\right] \leq \frac{1}{\gamma(1-\Theta)s}\frac{1}{T-T_0}\left(\frac{4L^2\sigma\sigma'_C}{\tau(1+\frac{1}{2}\gamma(1-\Theta)(T_0-t_0))}\right) + \frac{4L^2\sigma\sigma'_C s}{2\tau}$$
$$= \frac{4L^2\sigma\sigma'_C}{\tau}\left(\frac{1}{\gamma(1-\Theta)s}\frac{1}{T-T_0}\frac{1}{1+\frac{1}{2}\gamma(1-\Theta)(T_0-t_0)} + \frac{s}{2}\right).$$

Then, choosing $s = \frac{1}{(T-T_0)\gamma(1-\Theta)} \in [0,1]$ gives us

$$\mathbb{E}\left[G(\bar{\boldsymbol{\alpha}})\right] \leq \frac{4L^2\sigma\sigma'_C}{\tau}\left(\frac{1}{1+\frac{1}{2}\gamma(1-\Theta)(T_0-t_0)} + \frac{1}{(T-T_0)\gamma(1-\Theta)}\frac{1}{2}\right).$$

Therefore, by chosing T_0 and T such that the following inequalities hold, we obtain that $\mathbb{E}\left[G(\bar{\boldsymbol{\alpha}})\right] \leq \varepsilon_G$

$$\frac{4L^2\sigma\sigma'_C}{\tau}\left(\frac{1}{1+\frac{1}{2}\gamma(1-\Theta)(T_0-t_0)}\right) \leq \frac{1}{2}\varepsilon_G\,, \tag{2.29}$$
$$\frac{4L^2\sigma\sigma'_C}{\tau}\left(\frac{1}{(T-T_0)\gamma(1-\Theta)\frac{1}{2}}\right) \leq \frac{1}{2}\varepsilon_G\,. \tag{2.30}$$

Hence if

$$t_0 + \frac{2}{\gamma(1-\Theta)}\left(\frac{8L^2\sigma\sigma'_C}{\tau\varepsilon_G} - 1\right) \leq T_0\,,$$
$$T_0 + \frac{4L^2\sigma\sigma'_C}{\tau\varepsilon_G\gamma(1-\Theta)} \leq T\,. \qquad \square$$

then (2.29) and (2.30) are satisfied.

Chapter 3

Hessian-CoCoA

Motivated by the poor performances of CoCoA when applied to logistic regression cases, we have decided to investigate deeply the framework and find where eventually it could have been improved. Therefore, starting from an analogy between CoCoA framework (accurately described in the previous chapter), and the (block)-coordinate gradient descent method (see section 1.2.6 and, for more details, Tseng and Yun's paper [10]), the main idea behind Hessian-CoCoA is to incorporate more refined information about the curvature of the smooth function f in the local subproblems with the goal to achieve a faster convergence rate, and, at the same time, preserve communication efficiency. In this chapter we will present a new and more general local subproblem, and we will generalize the convergence analysis of CoCoA (see Section 2.3) to the new formulation. Finally, we will show that CoCoA can be derived as a special instance of this formulation.

3.1 Data-Local Subproblems

The goal of our new framework is to efficiently find a global minimizer of the objective (2.1) also for cases such as logistic regression, where the existing CoCoA framework was performing poorly. Indeed, the formulation of CoCoA's subproblems is based on an approximation of the Hessian matrix of f that results to be very distant from the real local curvature for most of the loss functions considered, resulting therefore in a bad local approximation of the original problem in each iteration. Again, we distribute the computation across the K machines that are storing the partitions $A_{[k]}$ of the data matrix A. Again we want to underline that, while distributing over the function g is straightforward, as we have required it to be separable in the assumptions, i.e. $g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} g_i(\alpha_i)$, the same does not hold for the smooth component $f(A\boldsymbol{\alpha})$: indeed, in order to split the minimization of this part of the objective function, we propose to minimize a block quadratic approxi-

mation of it, that, differently from (2.6), accounts for the exact curvature of f. In addition, for ERM problems, considering the exact $\nabla^2 f(A\boldsymbol{\alpha})$ in each of the local subproblems does not add extra communication costs: $\nabla^2 f(A\boldsymbol{\alpha})$ is either a scaled identity matrix, or it has a diagonal structure, and, in this last case, its elements can be easily derived from the shared vector $\mathbf{v}$ of global information. Consequently, the new framework maintains the same communication efficiency as CoCoA, but it improves the performances in terms of convergence rates, as we will show in Chapter 4.

Additional Assumption

Against this background, the new framework additionally requires f to be twice-differentiable, meaning that $\nabla^2 f(A\boldsymbol{\alpha})$ must exist.

Notation

To keep the notation shorter, we define $\ell = f \circ v$, where $v : \mathbb{R}^n \to \mathbb{R}^d$ is the affine map associated with the data matrix $A \in \mathbb{R}^{d \times n}$. As proved in lemma (1.17), the composed function ℓ is smooth and, by applying the chain rule, we obtain the following relations for its gradient and Hessian matrix:

$$\nabla \ell(\boldsymbol{\alpha}) = A^\top \nabla f(A\boldsymbol{\alpha}), \tag{3.1}$$

$$\nabla^2 \ell(\boldsymbol{\alpha}) = A^\top \nabla^2 f(A\boldsymbol{\alpha}) A. \tag{3.2}$$

Therefore, being $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{(t)}$ the variable computed in the iteration t, $\Delta\boldsymbol{\alpha}^{(t+1)} = \Delta\boldsymbol{\alpha}$ the descent direction that we are looking for at iteration $t+1$, and $\tilde{H}_t(\boldsymbol{\alpha}) = \tilde{H} \geq 0$ the approximation of $\nabla^2 \ell(\boldsymbol{\alpha})$ at iteration t, instead of the original problem (2.1), Hessian-CoCoA minimizes the following local approximation:

$$\Delta\boldsymbol{\alpha} = \min_{\Delta\boldsymbol{\alpha} \in \mathbb{R}^n} \nabla f(A\boldsymbol{\alpha})^\top A \Delta\boldsymbol{\alpha} + \frac{\sigma'_{HC}}{2} \Delta\boldsymbol{\alpha}^\top \tilde{H} \Delta\boldsymbol{\alpha} + g(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}), \tag{3.3}$$

and then computes the following update:

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \gamma \Delta\boldsymbol{\alpha}^{(t+1)}. \tag{3.4}$$

where $\gamma \in (0, 1]$ is the step length, and σ'_{HC} is chosen accordingly to (3.7).

Clearly, problem (3.3) has not solved the issue of distributing the computation, since $\tilde{H}$ could be a full matrix. We therefore propose a block-diagonal structure of the form

$$\tilde{H} = diag\left\{ A_{[1]}^\top \nabla^2 f(A\boldsymbol{\alpha}) A_{[1]}, \cdots, A_{[\bar{K}]}^\top \nabla^2 f(A\boldsymbol{\alpha}) A_{[\bar{K}]} \right\}, \tag{3.5}$$

where $\bar{K}$ is the number of blocks in $\tilde{H}$. In addition, for the distributed setting, we assume $\bar{K} = K$: in so doing, the local subproblem on machine k only requires access to the local data $A_{[k]}$, plus the vector $\mathbf{v} = A\boldsymbol{\alpha}$ of global information, that is shared across the machines at the beginning of each outer iteration. In particular, in each iteration the k machine optimizes the following local subproblem:

$$\mathcal{M}_k^{\sigma'_{HC}}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}) = \min_{\Delta\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^{n_k}} \nabla f(\mathbf{v})^\top A_{[k]} \Delta\boldsymbol{\alpha}_{[k]} + \frac{\sigma'_{HC}}{2} \Delta\boldsymbol{\alpha}_{[k]}^\top A_{[k]}^\top \nabla^2 f(\mathbf{v}) A_{[k]} \Delta\boldsymbol{\alpha}_{[k]} + \quad g(\boldsymbol{\alpha}_{[k]} + \Delta\boldsymbol{\alpha}_{[k]}) \,. \tag{3.6}$$

Examples of Configurations

- **one block per partition**
  If we assume that the blocks in $\tilde{H}$ are determined by how the matrix A is stored across K different machines (distributed case), then $\bar{K} = K$.
- **diagonal approximation**
  The main advantage in considering a diagonal structure for $\tilde{H}$ is that the local subproblems have a closed form solution that can be efficiently computed. Consequently, iterations get really computationally inexpensive even if suproblems are solved exactly ($\Theta = 1$). At the same time, less information are used locally, which results in an increase number of iterations required to achieve convergence, and,for a distributed environment, also in higher total communication costs. In this set-up, independently from the number K of machines, $\bar{K} = n$, where n is the number of columns of A.

3.2 Examples

Since the new framework has inherited from CoCoA both the mapping abstraction and the required assumptions (see Sections 2.1.1 and 2.1.2), the cases covered by Hessian-CoCoA are the same. The difference between the methods is in the local subproblems, and, in particular, how much they are able to capture the real curvature of f. In this section, we will show that for regularized ERM problems, $\nabla^2 f(A\boldsymbol{\alpha})$ has always a diagonal structure: indeed, it is either a scaled identity or, in the worst case, its elements can be easily recovered from the shared vector $\mathbf{v} = A\boldsymbol{\alpha}$ containing the global information. Consequently, this change does not affect the communication costs, but it preserves the efficiency of the original framework.

1. **Lasso**
   Given the presence of a non-smooth regularizer, the only possible choice is to map the problem to (2.1).

(A)-mapping:

$$f(A\boldsymbol{\alpha}) = \frac{1}{2}\|A\boldsymbol{\alpha} - \mathbf{b}\|^2 ,$$

and

$$\nabla^2 f(A\boldsymbol{\alpha})_{[i,j]} = \begin{cases} 1 & i = j \\ 0 & otherwise . \end{cases}$$

.

2. **Elastic Net Regression**
Both the mappings are possible.
(A)-mapping:

$$f(A\boldsymbol{\alpha}) = \frac{1}{2}\|A\boldsymbol{\alpha} - \mathbf{b}\|^2 ,$$

and

$$\nabla^2 f(A\boldsymbol{\alpha})_{[i,j]} = \begin{cases} 1 & i = j \\ 0 & otherwise . \end{cases}$$

.
(B)-mapping:

$$f^*(\mathbf{w}) = \eta\lambda\|\mathbf{w}\|_1 + (1-\eta)\frac{\lambda}{2}\|\mathbf{w}\|^2],.$$

Consequently,

$$f(A\boldsymbol{\alpha}) = \sum_{j=1}^{d} \frac{1}{2(1-\eta)\lambda}\left[(|\mathbf{y}_j^\top \boldsymbol{\alpha}| - \lambda\boldsymbol{\alpha})^+\right]^2 ,$$

and

$$\nabla^2 f(A\boldsymbol{\alpha})_{[i,j]} = \begin{cases} \frac{1}{2(1-\eta)\lambda}\frac{(|\mathbf{y}_j^\top \boldsymbol{\alpha} - \lambda\boldsymbol{\alpha}|)^+}{||\mathbf{y}_j^\top \boldsymbol{\alpha}| - \lambda\boldsymbol{\alpha}|} & i = j \\ 0 & otherwise . \end{cases}$$

3. **ℓ_2-Logistic Regression**
Both the mappings are possible.
(A)-mapping:

$$f(A\boldsymbol{\alpha}) = \sum_{j=1}^{d} \log(1 + \exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha})) ,$$

and

$$\nabla^2 f(A\boldsymbol{\alpha})_{[i,j]} = \begin{cases} \frac{\exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha})}{(1+\exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha}))^2} & i = j \\ 0 & otherwise . \end{cases}$$

(B)-mapping:

$$f^*(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 .$$

Consequently

$$f(A\boldsymbol{\alpha}) = \frac{1}{2\lambda}\|A\boldsymbol{\alpha}\|^2\,,$$

and

$$\nabla^2 f(A\boldsymbol{\alpha})_{[i,j]} = \begin{cases} \frac{1}{2\lambda} & i = j \\ 0 & \textit{otherwise}\,. \end{cases}$$

.

4. ℓ_1**-Logistic Regression**
   As for Lasso, the presence of the one norm regularizer constraints to map the problem exclusively to (2.1).
   (A)-mapping:

$$f(A\boldsymbol{\alpha}) = \sum_{j=1}^{d} \log(1 + \exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha}))\,,$$

   and

$$\nabla^2 f(A\boldsymbol{\alpha})_{[i,j]} = \begin{cases} \frac{\exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha})}{(1+\exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha}))^2} & i = j \\ 0 & \textit{otherwise}\,. \end{cases}$$

5. ℓ_2**-Support Vector Machine**
   The problem can only be mapped to (2.2), because the hinge loss is non-smooth.
   (B)-mapping:

$$f^*(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2\,.$$

   Consequently

$$f(A\boldsymbol{\alpha}) = \frac{1}{2\lambda}\|A\boldsymbol{\alpha}\|^2\,,$$

   and

$$\nabla^2 f(A\boldsymbol{\alpha})_{[i,j]} = \begin{cases} \frac{1}{2\lambda} & i = j \\ 0 & \textit{otherwise}\,. \end{cases}$$

   .

6. ℓ_1**-Support Vector Machine**
   The problem can not be adjusted in the framework: the same considerations done in the CoCoA for non smooth regularized SVM are valid for Hessian-CoCoA.

Algorithm 2 Distributed Hessian-CoCoA

Input: data matrix A distributed column-wise across K machines, aggregation parameter $\gamma \in (0, 1]$, and parameter σ'_{HC} for the local subproblems $\mathcal{M}_k^{\sigma'_{HC}}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v})$ as defined in (3.6)
central machine initialization $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{v}^{(0)} := \mathbf{0} \in \mathbb{R}^d$
set $t = -1$
while not converged **do**
  $t = t + 1$
  for $k \in \{1, 2, \dots, K\}$ **in parallel over machines do**
    compute a Θ-approximate solution $\Delta\boldsymbol{\alpha}_{[k]}$ of the local subproblem $\mathcal{M}_k^{\sigma'_{HC}}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}^{(t)})$
    update the local variable $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \gamma\Delta\boldsymbol{\alpha}_{[k]}$
    communicate to the central machine $\mathbf{v}_k = A_{[k]}\boldsymbol{\alpha}_{[k]}$
  end for
  central machine updates the global information $\mathbf{v}^{(t+1)} := \mathbf{v}^{(t)} + \gamma \sum_{k=1}^{K} \mathbf{v}_k$
end while

3.3 Convergence Analysis

This section is entirely dedicated to the analysis of Hessian-CoCoA convergence. The same lemmas and theorem of CoCoA are proposed again for this new framework, but small changes are made in order to genelarize their validity to the new subproblem formulation.

3.3.1 σ'_{HC} Parameter

As σ'_C for CoCoA, the role of the σ'_{HC} parameter is foundamental for convergence. In this section we will give a formal definition of this parameter, and, by introducing the additional assumption of f being self-concordant like (see definition 1.20), we will provide a safe and data independent criterion to tune it. In addition, we will recover the old σ'_C definition of the CoCoA framework as a special case.

Definition 3.1 *Given a step size $\gamma \in (0, 1]$, the value of the σ'_{HC} parameter should be chosen accordingly to the following inequlity*

$$\sigma'_{HC} \geq (\sigma'_{HC})_{min} := \gamma \max_{\Delta\boldsymbol{\alpha} \in \mathbb{R}^n} \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \max_{\theta \in [0,1]} \frac{\Delta\boldsymbol{\alpha}^\top \nabla^2 \ell(\boldsymbol{\alpha} + \theta\Delta\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}}{\sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}_{[k]}}, \tag{3.7}$$

where $\tilde{H}(\boldsymbol{\alpha})$ is defined as in (3.5).

By exploiting the block structure of the Hessian approximation matrix $\tilde{H}(\boldsymbol{\alpha})$, we obtain

$$\begin{aligned}\sigma'_{HC} \geq (\sigma'_{HC})_{min} &:= \gamma \max_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \max_{\theta\in[0,1]} \frac{\langle A\Delta\boldsymbol{\alpha}, A\Delta\boldsymbol{\alpha}\rangle_{\nabla^2 f(A(\boldsymbol{\alpha}+\theta\Delta\boldsymbol{\alpha}))}}{\sum_{\bar{k}=1}^{\bar{K}} \langle A_{[\bar{k}]}\Delta\boldsymbol{\alpha}_{[\bar{k}]}, A_{[\bar{k}]}\Delta\boldsymbol{\alpha}_{[\bar{k}]}\rangle_{\nabla^2 f(A\boldsymbol{\alpha})}} \\ &= \gamma \max_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \max_{\theta\in[0,1]} \frac{\langle \Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}\rangle_{\nabla^2 \ell(\boldsymbol{\alpha}+\theta\Delta\boldsymbol{\alpha})}}{\sum_{\bar{k}=1}^{\bar{K}} \langle \Delta\boldsymbol{\alpha}_{[\bar{k}]}, \Delta\boldsymbol{\alpha}_{[\bar{k}]}\rangle_{\tilde{H}(\boldsymbol{\alpha})}} \end{aligned} \tag{3.8}$$

where $\bar{K}$ is the number of blocks. In addition, if the blocks are determined by the number of partitions, then $K = \bar{K}$.

For the case of logistic regression, it is possible to recover a safe data-independent bound on the value of this parameter, that allows our framework to achieve convergence without introducing any additional assumption on the structure of the data matrix A.

Given definition (3.8), and assuming ℓ to be self-concordant like (see definition 1.20), we derive the following bounds on $(\sigma'_{HC})_{min}$.

Lemma 3.2 *(Safe Data-Independent Bound). Assuming ℓ to be self-concordant like with constant M_ℓ, the following bound on σ'_{HC} preserves the convergence*

$$(\sigma'_{HC})_{min} \leq \tilde{\sigma} := \gamma \max_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \exp(M_\ell \|\Delta\boldsymbol{\alpha}\|_2)\bar{K}\,, \tag{3.9}$$

where $\bar{K}$ is equal to the number of blocks in $\tilde{H}$.

Proof Given that $\Delta\boldsymbol{\alpha} = \Delta\boldsymbol{\alpha}^{(t)}$, $\Delta\mathbf{v} = A\Delta\boldsymbol{\alpha}$ and $\Delta\mathbf{v}_{\bar{k}} = A_{[\bar{k}]}\Delta\boldsymbol{\alpha}_{[\bar{k}]}$, we begin by applying the definition of $(\sigma'_{HC})_{min}$ with $\tilde{H}(\boldsymbol{\alpha})$ defined as in equation (3.5), and considering the property of self-concordant like functions (see proposition 1.21):

$$\begin{aligned}(\sigma'_{HC})_{min} &:= \gamma \max_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \max_{\theta\in[0,1]} \frac{\Delta\boldsymbol{\alpha}^\top \nabla^2 \ell(\boldsymbol{\alpha}+\theta\Delta\boldsymbol{\alpha})\boldsymbol{\alpha}}{\sum_{\bar{k}=1}^{\bar{K}} \Delta\boldsymbol{\alpha}_{[\bar{k}]}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}_{[\bar{k}]}} \\ &\leq \gamma \max_{\Delta\mathbf{v}\in\mathbb{R}^d} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \exp(M_\ell\|\Delta\boldsymbol{\alpha}\|_2) \frac{\Delta\mathbf{v}^\top \nabla^2 f(A\boldsymbol{\alpha})\Delta\mathbf{v}}{\sum_{\bar{k}=1}^{\bar{K}} \Delta\mathbf{v}_{\bar{k}}^\top \nabla^2 f(A\boldsymbol{\alpha})\Delta\mathbf{v}_{\bar{k}}}\end{aligned}$$

We then proceed by defining the convex function $h : \mathbb{R}^d \to \mathbb{R}$, $h(\mathbf{u}) = \mathbf{u}^\top \nabla^2 f(A\boldsymbol{\alpha})\mathbf{u}$ and by applying the Jensen's inequality to it

$$
\begin{aligned}
(\sigma'_{HC})_{min} &\leq \gamma \max_{\Delta \mathbf{v} \in \mathbb{R}^d} \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \exp(M_\ell \|\Delta \boldsymbol{\alpha}\|_2) \frac{\Delta \mathbf{v}^\top \nabla^2 f(A\boldsymbol{\alpha}) \Delta \mathbf{v}}{\bar{K} \sum_{\bar{k}=1}^{\bar{K}} \frac{1}{\bar{K}} h(\Delta \mathbf{v}_{\bar{k}})} \\
&\leq \gamma \max_{\Delta \mathbf{v} \in \mathbb{R}^n} \exp(M_\ell \|\Delta \boldsymbol{\alpha}\|_2) \frac{\Delta \mathbf{v}^\top \nabla^2 f(A\boldsymbol{\alpha}) \Delta \mathbf{v}}{\frac{1}{\bar{K}} h(\sum_{\bar{k}=1}^{\bar{K}} \Delta \mathbf{v}_{\bar{k}})} \\
&= \gamma \max_{\Delta \boldsymbol{\alpha} \in \mathbb{R}^n} \exp(M_f \|\Delta \boldsymbol{\alpha}\|_2) \bar{K} \,. \qquad \square
\end{aligned}
$$

In addition, lemma 3.2 combined with the L-bounded support modification of g (see section 2.2.4) leads to the final corollary.

Corollary 3.3 *Assuming ℓ to be self-concordant like with constant M_ℓ, g_i to have an L-bounded support $\forall i \in [n]$, the following bound on $\tilde{\sigma}$ can be derived*

$$\tilde{\sigma} \leq \gamma \exp(2\sqrt{n} M_\ell L) \bar{K} \,, \tag{3.10}$$

where $\bar{K}$ is equal to the number of blocks in $\tilde{H}$.

Proof The proof follows trivially by applying the definition of L-bounded support to equation (3.9). $\square$

Note

For quadratic functions, $M_\ell = 0$.

Logistic Regression Case

Following the same line of Francis Bach in his paper [1], for logistic regression the bound (3.10) reduces to

$$\tilde{\sigma} \leq \gamma \exp(2\sqrt{n} L) \bar{K} \,. \tag{3.11}$$

Indeed, considering

$$\ell(\boldsymbol{\alpha}) = \sum_{j=1}^{d} \log(1 + \exp(-b_i \mathbf{y}_i^\top \boldsymbol{\alpha})) \,,$$

we obtain that M_ℓ is nothing but the maximum ℓ_2-norm of all data points

$$M_\ell = \max_{j \in \{1,\dots,d\}} \|\mathbf{y}_j\|_2 \,. \tag{3.12}$$

Assuming a normalization of the data, we can easily upper bound it with 1.

3.3.2 Lemmas and Theorem of Convergence

Regarding the convergence analysis, we will repeat the same steps as in CoCoA: we will start with three foundamental lemmas, whose results will lead to the final theorem. To avoid repetitions, only the parts which needed to be modified will be exposed, the others are exactly identical as in the CoCoA analysis.

Note1

Without loss of generality, in order to be coherent with the notation used in the CoCoA convergence analysis (see Section 2.3), we will assume to work with $\bar{K} = K$, meaning that the blocks of $\tilde{H}$ are determined by the partitions of the data matrix A across different machines. Consequently:

$$\tilde{H} := \tilde{H}_t = diag\left\{ A_{[1]}^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A_{[1]}, \cdots, A_{[K]}^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A_{[K]} \right\} \quad \forall t\,.$$

The proof is anyway valid for any general choice of $\bar{K}$.

Note2

The results of the following lemmas and theorem are valid for any choice of σ'_{HC} such that equation (3.7) holds. In addition, all the working assumptions of Hessian-CoCoA framework regarding f and g are required to be fulfilled. In particular, f is required to be $\frac{1}{\tau}$-smooth, and twice differentiable, and g is required to be convex, separable and L-bounded.

Notation

For simplicity, we will drop the apex t referring to the iteration, unless really necessary to avoid wrong interpretations.

Lemma 3.4 *For any $\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha} \in \mathbb{R}^n, \mathbf{v} = \mathbf{v}(\boldsymbol{\alpha}) := A\boldsymbol{\alpha}$, $\gamma \in (0, 1]$, it holds that*

$$\mathcal{O}_A(\boldsymbol{\alpha} + \gamma \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]}) \leq (1-\gamma)\mathcal{O}_A(\boldsymbol{\alpha}) + \gamma \sum_{k=1}^{K} \mathcal{M}_k^{\sigma'_{HC}}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v})\,. \quad (3.13)$$

Proof In this proof, the assumption of $(\frac{1}{\tau})$-smoothness of f is foundamental. After one outer iteration, Hessian-CoCoA performs on the objective $\mathcal{O}_A(\cdot)$ the following update:

$$\mathcal{O}_A(\boldsymbol{\alpha} + \gamma \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]}) = \underbrace{f(\mathbf{v}(\boldsymbol{\alpha} + \gamma \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]}))}_{\text{A}} + \underbrace{\sum_{i=1}^{n} g_i(\alpha_i + \gamma(\sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]})_i)}_{\text{B}}\,. \quad (3.14)$$

We proceed by analyzing separately the terms A and B, and by finding a bound for them.

A term
The bound on A is found by considering the $(\frac{1}{\tau})$-smoothness of f, together with its second order Taylor approximation:

$$
\begin{aligned}
A &= f(\mathbf{v}(\boldsymbol{\alpha} + \gamma \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]})) = f(\mathbf{v}(\boldsymbol{\alpha}) + \gamma \sum_{k=1}^{K} \mathbf{v}(\Delta\boldsymbol{\alpha}_{[k]})) \\
&\overset{\text{Prop. 1.22}}{=} f(\mathbf{v}(\boldsymbol{\alpha})) + \gamma \nabla f(\mathbf{v}(\boldsymbol{\alpha}))^\top \mathbf{v}(\Delta\boldsymbol{\alpha}) + \frac{\gamma^2}{2} \mathbf{v}(\Delta\boldsymbol{\alpha})^\top \nabla^2 f(\mathbf{v}(\boldsymbol{\alpha} + \theta\Delta\boldsymbol{\alpha})) \mathbf{v}(\Delta\boldsymbol{\alpha}) \\
&\overset{\text{definition of } \mathbf{v}}{=} f(\mathbf{v}(\boldsymbol{\alpha})) + \gamma \nabla f(\mathbf{v}(\boldsymbol{\alpha}))^\top A\Delta\boldsymbol{\alpha} + \frac{\gamma^2}{2} \Delta\boldsymbol{\alpha}^\top A^\top \nabla^2 f(\mathbf{v}(\boldsymbol{\alpha} + \theta\Delta\boldsymbol{\alpha})) A\Delta\boldsymbol{\alpha} \\
&\overset{\text{safe value } \sigma'_{HC}}{\leq} f(\mathbf{v}(\boldsymbol{\alpha})) + \gamma \nabla f(\mathbf{v}(\boldsymbol{\alpha}))^\top A\Delta\boldsymbol{\alpha} + \frac{\gamma\sigma'_{HC}}{2} \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]}^\top \tilde{H}(\boldsymbol{\alpha}) \Delta\boldsymbol{\alpha}_{[k]} \,.
\end{aligned}
$$

The rest of the proof follows exactly the same steps as the corresponding part of lemma 2.4. □

Lemma 3.5 *Let g_i be convex. Then for all iterations of Hessian-CoCoA under the assumption 2.1 for the local solvers, and any $s \in [0, 1]$, it holds that*

$$
\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] \geq \gamma(1-\Theta)\left(sG(\boldsymbol{\alpha}^{(t)}) - \frac{\sigma'_{HC} s^2}{2} R^{(t)}\right), \quad (3.15)
$$

where

$$
R^{(t)} := (\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}), \quad (3.16)
$$

for $\mathbf{u}^{(t)} \in \mathbb{R}^n$ with

$$
u_i^{(t)} \in \partial g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}^{(t)})) \,. \quad (3.17)
$$

Proof By applying lemma (3.4), we obtain that:

$$
\begin{aligned}
\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] &= \mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}) - \mathcal{O}_A(\boldsymbol{\alpha} + \gamma \sum_{k=1}^{K} \Delta\boldsymbol{\alpha}_{[k]})\right] \\
&\geq \gamma(1-\Theta)\left(\underbrace{\mathcal{O}_A(\boldsymbol{\alpha}) - \sum_{k=1}^{K} \mathcal{M}_k^{\sigma'_{HC}}(\Delta\boldsymbol{\alpha}^\star_{[k]}; A_{[k]}, \mathbf{v})}_{\mathrm{C}}\right) .
\end{aligned} \quad (3.18)
$$

Now, we look for a bound on C by applying Jensen's inequality to the functions g_i, and by substituting $\Delta\boldsymbol{\alpha}_i^\star$ with the subgradient $s(u_i - \alpha_i)$, where u_i satifies the optimality condition by assumption. This gives us:

$$\begin{aligned} C &= \sum_{i=1}^{n}(g_i(\alpha_i) - g_i(\alpha_i + \Delta\boldsymbol{\alpha}_i^\star)) - (A\Delta\boldsymbol{\alpha}^\star)^\top \mathbf{w}(\boldsymbol{\alpha}) - \frac{\sigma'_{HC}}{2}\Delta\boldsymbol{\alpha}^{\star\top}\tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}^\star \\ &= \sum_{i=1}^{n}(sg_i(\alpha_i) - sg_i(u_i)) - A(s(\mathbf{u}-\boldsymbol{\alpha}))^\top \mathbf{w}(\boldsymbol{\alpha}) - \frac{s^2\sigma'_{HC}}{2}(\mathbf{u}-\boldsymbol{\alpha})^\top\tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha})\,. \end{aligned} \tag{3.19}$$

We next analyze the duality gap, and by plugging in the definition of problem (2.1) and problem (2.2), and by exploiting the definition of convex conjugate functions (1.1), we obtain:

$$\begin{aligned} G(\boldsymbol{\alpha}) := \mathcal{O}_A(\boldsymbol{\alpha}) - \mathcal{O}_B(\mathbf{w}(\boldsymbol{\alpha})) &= \sum_{i=1}^{n}\left(g_i^*(-\mathbf{x}_i^\top\mathbf{w}(\boldsymbol{\alpha})) + g_i(\alpha_i)\right) + f^*(\nabla f(A\boldsymbol{\alpha})) + f(A\boldsymbol{\alpha}) \\ &= \sum_{i=1}^{n}\left(g_i^*(-\mathbf{x}_i^\top\mathbf{w}(\boldsymbol{\alpha})) + g_i(\alpha_i) + \alpha_i\mathbf{x}i^\top\mathbf{w}(\boldsymbol{\alpha})\right)\,. \end{aligned}$$

From the maximal property of convex conjugate functions, we have that:

$$g_i(u_i) = u_i(-\mathbf{x}_i^\top\mathbf{w}(\boldsymbol{\alpha})) - g_i^*(-\mathbf{x}_i^\top\mathbf{w}(\boldsymbol{\alpha}))\,, \tag{3.20}$$

and, consequently, by plugging (3.20) back into (3.19) we obtain:

$$\begin{aligned} C &\geq \sum_{i=1}^{n}\left(sg_i(\alpha_i) - su_i(-\mathbf{x}_i^\top\mathbf{w}(\boldsymbol{\alpha})) + sg_i^*(-\mathbf{x}_i^\top\mathbf{w}(\boldsymbol{\alpha}))\right) \\ &\quad - A(s(\mathbf{u}-\boldsymbol{\alpha}))^\top\mathbf{w}(\boldsymbol{\alpha}) - \frac{s^2\sigma'_{HC}}{2}(\mathbf{u}-\boldsymbol{\alpha})^\top\tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha}) \\ &= sG(\boldsymbol{\alpha}) - \frac{s^2\sigma'_{HC}}{2}(\mathbf{u}-\boldsymbol{\alpha})^\top\tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha})\,. \end{aligned} \tag{3.21}$$

□

The conclusion follows straightforwad, by plugging (3.21) back into (3.18).

The following lemma is proving that the $R^{(t)}$ term is bounded above, assuming g_i^* being L-Lipschitz continuous $\forall i \in [n]$, or, equivalently, that g_i have an L-bounded support $\forall i \in [n]$.

Lemma 3.6 *If g_i^* are L-Lipschitz continuous for all $i \in [n]$, then*

$$R^{(t)} \leq \frac{4L^2}{\tau} \underbrace{\sum_{k=1}^{K} \sigma_k n_k}_{=:\sigma} \quad \forall t\,, \tag{3.22}$$

where σ_k is nothing but the induced norm of $A_{[k]}$

$$\sigma_k := \max_{\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^n} \frac{\|A_{[k]}\boldsymbol{\alpha}_{[k]}\|^2}{\|\boldsymbol{\alpha}_{[k]}\|^2} \quad \forall k \in [K]\,. \tag{3.23}$$

Proof The proof is mainly based on the fact that, by assuming g_i^* to be L-Lipschitz continuous, we are also assuming g_i to have an L-bounded support, since, given that g_i and g_i^* are convex conjugate pairs, these requirements are equivalent.

$$\begin{aligned}
R^{(t)} &= (\mathbf{u}-\boldsymbol{\alpha})^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha}) = \sum_{k=1}^{K} (\mathbf{u}-\boldsymbol{\alpha})_{[k]}^\top A_{[k]}^\top \nabla^2 f(A\boldsymbol{\alpha}) A_{[k]} (\mathbf{u}-\boldsymbol{\alpha})_{[k]} \\
&\leq \frac{1}{\tau} \sum_{k=1}^{K} (\mathbf{u}-\boldsymbol{\alpha})_{[k]}^\top A_{[k]}^\top A_{[k]} (\mathbf{u}-\boldsymbol{\alpha})_{[k]} = \frac{1}{\tau} \sum_{k=1}^{K} \|A_{[k]}(\mathbf{u}^{(t)}-\boldsymbol{\alpha}^{(t)})_{[k]}\|^2 \\
&\leq \frac{1}{\tau} \sum_{k=1}^{K} \|A_{[k]}\|^2 \|(\mathbf{u}^{(t)}-\boldsymbol{\alpha}^{(t)})_{[k]}\|^2 = \frac{1}{\tau} \sum_{k=1}^{K} \sigma_k \|(\mathbf{u}^{(t)}-\boldsymbol{\alpha}^{(t)})_{[k]}\|^2 \\
&\leq \frac{4L^2}{\tau} \sum_{k=1}^{K} \sigma_k n_k
\end{aligned}$$

□

The following theorem is using the results proven in the previous lemmas to demonstrate that Hessian-CoCoA is achieving convergence to the optimal solution of problem (2.1) by solving with an arbitrary accuracy the local subproblems (3.6).

Theorem 3.7 *Consider Algorithm 2, and a local solver of quality Θ. Let $g_i^*(\cdot)$ be L-Lipschitz continuous, and $\varepsilon_G \geq 0$ be the desired duality gap value. Then, after T iterations, where*

$$\begin{aligned}
T &\geq T_0 + \max\left\{ \left\lceil \frac{1}{\gamma(1-\Theta)} \right\rceil, \left\lceil \frac{4L^2\sigma\sigma'_{HC}}{\tau\varepsilon_G\gamma(1-\Theta)} \right\rceil \right\} \\
T_0 &\geq t_0 + \left\lceil \left[\frac{2}{\gamma(1-\Theta)} \left(\frac{8L^2\sigma\sigma'_{HC}}{\tau\varepsilon_G} - 1 \right) \right]^+ \right\rceil \\
t_0 &\geq \max\left(0\ ,\ \left\lceil \frac{1}{\gamma(1-\Theta)} \log\left(\frac{\tau(\mathcal{O}_A(\boldsymbol{\alpha}^{(0)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star))}{2L^2\sigma\sigma'_{HC}} \right) \right\rceil \right)
\end{aligned}$$

we have that the expected duality gap satifies

$$\mathbb{E}\left[\mathcal{O}_A(\bar{\boldsymbol{\alpha}}) - (-\mathcal{O}_B(\mathbf{w}(\bar{\boldsymbol{\alpha}})))\right] \geq \varepsilon_G$$

where $\bar{\boldsymbol{\alpha}}$ *is the averaged iterate*

$$\bar{\boldsymbol{\alpha}} := \frac{1}{T - T_0} \sum_{t=T_0}^{T-1} \boldsymbol{\alpha}^{(t)} .$$

Proof The proof is equal to the one exposed for CoCoA. □

3.3.3 Hessian-CoCoA and CoCoA Comparison for Logistic Regression

Being $\bar{K} = K$ with K the number of partitions, and given the same safe value δ for σ'_{HC} and σ'_C parameters, Hessian-CoCoA always performs better than CoCoA, since in each iteration it uses a tighter local approximation of the objective function. It is well known that, for $\Delta\boldsymbol{\alpha}$ sufficiently small, among all the possible second order approximations of a multivariate function, the best model is given by its second order Taylor expansion. Starting from this last consideration and analysing the case of logistic regression loss, we will show that the new framework ensures a stronger convergece rate, by respectively comparing Hessian-CoCoA and CoCoA with the second order Taylor expansion of ℓ about the current iterate $\boldsymbol{\alpha}$.
Wlog, we assume to work with $n = N$, and K partitions $\mathcal{P}_k \ \forall k \in [K]$. In this case, the second order Taylor expansion of $\ell = f \circ v$ about $\boldsymbol{\alpha}$ is

$$T_{2,\boldsymbol{\alpha}}(\Delta\boldsymbol{\alpha}) := \ell(\boldsymbol{\alpha}) + (\nabla f(A\boldsymbol{\alpha}))^\top A\Delta\boldsymbol{\alpha} + \frac{1}{2}\Delta\boldsymbol{\alpha}^\top A^\top \nabla^2 f(A\boldsymbol{\alpha}) A\Delta\boldsymbol{\alpha} .$$

By the reminder theorem, we have that:

$$|\ell(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) - T_{2,\boldsymbol{\alpha}}(\Delta\boldsymbol{\alpha})| = |R_2| ,$$

where R_2 is the remainder term in the second order Taylor expansion. In particular, for self-concordant like functions as the logistic regression, R_2 can be upper bounded by combining proposition 1.21 together with definition 1.20 (see [1] for more details).

The distance between the second order Taylor expansion and CoCoA subproblem about $\boldsymbol{\alpha}$ is given by

$$|T_{2,\boldsymbol{\alpha}}(\Delta\boldsymbol{\alpha}) - \sum_{k\in[K]} \mathcal{G}_k^\delta(\Delta\boldsymbol{\alpha}_{[k]})| := |\sum_{k\in K} (A_{[k]}^\top \nabla^2 f(A\boldsymbol{\alpha}) A_{[k]}^\top - \delta A_{[k]}^\top A_{[k]})\Delta\boldsymbol{\alpha}_{[k]}^2 + \sum_{k\in K} C_k| ,$$

and the distance between the second order Taylor expansion and Hessian-CoCoA subproblem about $\boldsymbol{\alpha}$ is given by

$$|T_{2,\boldsymbol{\alpha}}(\Delta\boldsymbol{\alpha}) - \sum_{k\in[K]} \mathcal{M}_k^{\delta}(\Delta\boldsymbol{\alpha}_{[k]})| := |\sum_{k\in K}(A_{[k]}^{\top}\nabla^2 f(A\boldsymbol{\alpha})A_{[k]}^{\top} - \delta A_{[k]}^{\top}\nabla^2 f(A\boldsymbol{\alpha})A_{[k]})\Delta\boldsymbol{\alpha}_{[k]}^2 + \sum_{k\in K} C_k|\,,$$

where

$$C_k = \sum_{i\in\mathcal{P}_k}\sum_{j\notin\mathcal{P}_k} A_{i,j}A_{j,i}\nabla^2_{i,j} f(A\boldsymbol{\alpha})\Delta\alpha_i\Delta\alpha_j \quad \forall k \in [K]\,,$$

and where $\mathcal{G}_k^{\delta}$ and $\mathcal{M}_k^{\delta}$ indicates the local subproblems for the k partition for CoCoA and Hessian-CoCoA respectively.

Since for logistic regression $\nabla^2 f(A\boldsymbol{\alpha}) \leq \frac{1}{4}I$ and $\delta > \frac{\delta}{4}$, we can conclude that Hessian-CoCoA subproblem is a tighter local approximation than the one used in CoCoA, i.e. $|T_{2,\boldsymbol{\alpha}}(\Delta\boldsymbol{\alpha}) - \mathcal{M}^{\delta}(\Delta\boldsymbol{\alpha})| < |T_{2,\boldsymbol{\alpha}}(\Delta\boldsymbol{\alpha}) - \mathcal{G}^{\delta}(\Delta\boldsymbol{\alpha})|$, and, therefore, Hessian-CoCoA has a better global convergence rate given $\sigma'_{HC} = \sigma'_C = \delta$, with δ safe value.

3.4 Another Perspective on CoCoA

Hessian-CoCoA combines the strategy of a block-coordinate gradient descent method (see section 1.2.6 in the Introduction), with the possibility of solving the local subproblems to an arbitrary accuracy, as required in assumption 2.1. Indeed, differently from a block-coordinate gradient descent method, our framework does not require the exact solution of the local subproblems, but the time spent in solving the local subproblems, and therefore the accuracy of the local solutions, depends on the application and, in the distributed setting, should also be tuned according to the communication costs. Both CoCoA and Hessian-CoCoA can be interpreted as a generalization of block-coordinate gradient descent method: in particular, regarding Hessian-CoCoA, $\tilde{H}_t$ is given by equation (3.5); while, in the CoCoA framework, the local subproblems do not account for the exact curvature of f, but $\nabla^2 f(A\boldsymbol{\alpha})$ is crudely approximated with the upper bound $\frac{1}{\tau}I$ coming from the smoothness of f.

σ'_C parameter

Also the CoCoA definition of the σ'_C parameter can be derived from Hessian-CoCoA definition (3.8) by replacing $\nabla^2 f(A\boldsymbol{\alpha})$ with $\frac{1}{\tau}I$. In particular, assuming $K = \bar{K}$:

$$\sigma'_C \geq (\sigma'_C)_{min} := \gamma \max_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{\langle A\Delta\boldsymbol{\alpha}, A\Delta\boldsymbol{\alpha}\rangle_{\frac{1}{\tau}I}}{\sum_{k=1}^{K}\langle A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}, A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}\rangle_{\frac{1}{\tau}I}}\,, \tag{3.24}$$

Least Squares Loss Case

The two frameworks are closely related and for the case $f(A\boldsymbol{\alpha}) = \frac{1}{2}\|A\boldsymbol{\alpha} - \mathbf{b}\|^2$ they even coincides. The least squares loss functon is indeed 1-smooth and $\nabla^2 f(A\boldsymbol{\alpha}) = I$, therefore, assuming a block diagonal structure where $\bar{K} = K$, the local subproblems of the two frameworks are exactly the same, since $\tilde{H} = \sum_{k=1}^{K} A_{[k]}^\top A_{[k]}$. Similar considerations regarding the bound (3.10) could be drawn by noticing that $M_\ell = 0$ for quadratic functions: indeed, this allows to recover from equation (3.10) the same safe data-independent bound valid for σ'_C.

Chapter 4

Experiments

4.1 Running Example

Since the main reason behind Hessian-CoCoA is to improve the performances of CoCoA in cases such as logistic regression loss function, where the existing framework is performing poorly, we have chosen as running example ℓ_1-logistic regression. Logistic regression with one norm regularizer is widely used in large scale machine learning applications: indeed, the non-smooth regularizer not only helps against over-fitting, but it induces sparsity in the solution and, consequently, it is often used for feature selection. As already mentioned, the presence of the non smooth regularizer only allows for a mapping to (2.1). Therefore, both CoCoA and Hessian-CoCoA act as primal methods, and the data matrix results to be distributed by features. The regularized ERM problem addressed in our experiments is

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad \sum_{j=1}^{d} \log(1+\exp(-b_i \mathbf{y}_j^\top \boldsymbol{\alpha})) + \lambda\|\boldsymbol{\alpha}\|_1\,, \tag{4.1}$$

where $\mathbf{b} \in \{-1,1\}^d$ is the vector containing the target variables, $A = [\mathbf{y}_1^\top, \ldots, \mathbf{y}_d^\top] \in \mathbb{R}^{d\times n}$ is the data matrix, where n is the number of features and d the number of examples, $\mathbf{y}_j \in \mathbb{R}^n$ is the vector of the input variables for the example j, and $\lambda > 0$ is the regularization parameter.

4.1.1 Primal Formulation

In order to adjust (4.1) in the CoCoA and Hessian-CoCoA framework, the problem is mapped to (2.1) with an L-bound modification of the regularizer, resulting in the following formulation

$$\min_{\substack{\boldsymbol{\alpha}\in\mathbb{R}^n \\ \alpha_i\in[-L,L]}} \quad \sum_{j=1}^{d} \log(1+\exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha})) + \lambda\|\boldsymbol{\alpha}\|_1\,. \tag{4.2}$$

4.1.2 Dual Formulation

In order to compute the duality gap, that is used to approximately evaluate the suboptimality of the current solution, and, therefore, as a stopping condition for the algorithm, we need to derive the dual formulation (2.2) in the specific case where the primal (2.1) has the form of (4.2). In particular, the dual problem is

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad B\left[\|-A^\top\mathbf{w}\|_\infty-\lambda\right]^+ + f^*(\mathbf{w})\,,$$

where

$$f^*(\mathbf{w}) := \begin{cases} \sum\limits_{\substack{j=1\\ w_j>0}}^{d} (-b_jw_j)\log(1-b_jw_j) + \sum\limits_{\substack{j=1\\ w_j<1}}^{d} (1+b_jw_j)\log(1+b_jw_j) & -b_jw_j\in[0,1]\ \ \forall j\in[d] \\ -\infty & \text{otherwise}\,. \end{cases}$$

4.2 Local Solvers

The local subproblems both in CoCoA (2.7) and Hessian-CoCoA (3.6), are not trivial to be solved. Their complexity depends mainly on their size and on the structure of the local Hessian matrix. Motivated by the great success of coordinate gradient descent method in the late big scale machine learning applications, we have decided to use it as local solver. In particular, we have implemented coordinate descent method with random permutation principle over a subset P of the local optimization variables with a non-zero gradient.

Algorithm

Algorithm 3 Coordinate Descent with Random Permutation

Input: local column-partition $A_{[k]} \in \mathbb{R}^{d \times n_k}$, parameter σ'_C for CoCoA, and σ'_{HC} Hessian-CoCoA, global vector $\mathbf{v} \in \mathbf{R}^d$, number of rounds N_{max}, local variable $\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^{n_k}$
initialization $\Delta\boldsymbol{\alpha}_{[k]} = \mathbf{0}$
randomly select a subset $\mathcal{J}_k$ of indices such that $\mathcal{J}_k \subset \{1, \dots, n_k\}, |\mathcal{J}_k| = P, \forall j \in \mathcal{J}_k \ : \ \left(\nabla f(\mathbf{v})^\top A_{[k]}\right)_j \neq 0, \forall h, j \in \mathcal{J}_k \ h \neq j$
for $i = 1, \dots, N_{max}$ **do**
  for $j \in J_k$ **do**
    fix all the components except for the j one: $\Delta\boldsymbol{\alpha}_{[k]} = [\Delta\bar{\alpha}_1, \dots, \Delta\bar{\alpha}_{j-1}, \Delta\alpha_j, \Delta\bar{\alpha}_{j+1}, \dots, \Delta\bar{\alpha}_{n_k}]$
    solve $\Delta\alpha_j = \min_{\Delta\alpha_j} \mathcal{G}_k^{\sigma'_C}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}) =$ (2.7) for CoCoA, and $\Delta\alpha_j = \min_{\Delta\alpha_j} \mathcal{M}_k^{\sigma'_{HC}}(\Delta\boldsymbol{\alpha}_{[k]}; A_{[k]}, \mathbf{v}) =$ (3.6) for Hessian-CoCoA
  end for
end for
communicate to the central machine $\mathbf{v}_k = A_{[k]}(\boldsymbol{\alpha}_{[k]} + \Delta\boldsymbol{\alpha}_{[k]})$

Note

The size of P and the number of rounds N_{max} should be tuned accordingly to the application.

4.3 Results

Two different configurations have been considered for the experiments, which are meaningful in two opposite cases.

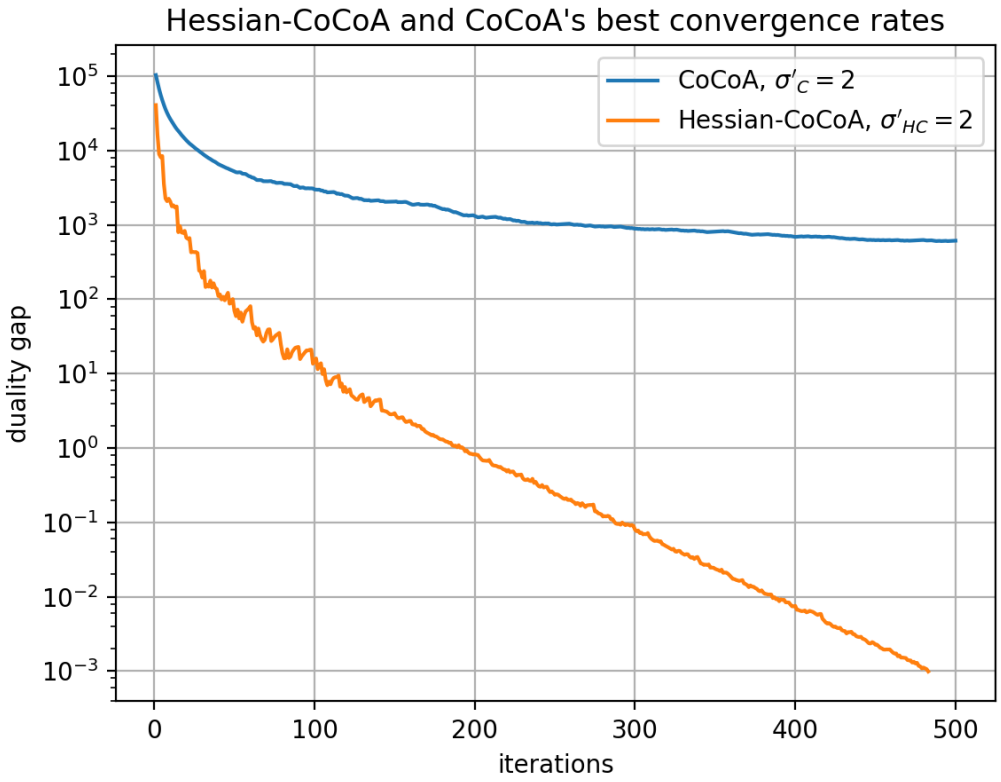
First Configuration

In the first configuration, we consider to operate in a distributed setting with two machines. Each machine stores a balanced partition of the data matrix $A \in \mathbb{R}^{d \times n}$: in particular, given n_1 and n_2 to be the size of each partition, where $n_1 = n_2$, $n_1 + n_2 = n$; machine 1 stores $A_{[1]} \in \mathbb{R}^{d \times n_1}$, and machine 2 stores $A_{[2]} \in \mathbb{R}^{d \times n_2}$. In this configuration, we have assumed communication costs to be significant, as it typically happens in a distributed setting. Consequently, we primarily aim at descreasing the total number of iterations, by including more information in the local subproblems through the local hessian approximation, i.e. respectively $\sum_{k=1}^K A_{[k]}^\top A_{[k]}$ for CoCoA,

and $\sum_{k=1}^{K} A_{[k]}^{\top} \nabla^2 f(A\boldsymbol{\alpha}) A_{[k]}$ for Hessian-CoCoA, and by approximately solving them. In so doing the resulting number of total iterations decreases significantly with respect to the extreme case of considering a local approximation of the Hessian matrix with a diagonal structure. On the other hand, the computation cost of each iteration significantly increases.

Second Configuration

Regarding the second configuration, it corresponds to the dual situation with respect to the one analyzed before: in particular, we assume the communication costs to be not relevant. This assumption is not very much typical of a distributed setting, but it is characteristic of a parallel framework, where the data matrix A is stored in a single memory that is shared between the multiple processors that are executing the computation. Given these assumptions, we conversely aim at decreasing the total time spent for the optimization, meaning the time spent in each single iteration. To achieve this goal, independently from the number of machines/processors taking part in the computation, we assume a local diagonal approximation of the Hessian matrix for both CoCoA and Hessian-CoCoA. In so doing, the local subproblems have a closed form solution, resulting in an incredible speed up of the iterations, and, conversely, in increasing their total number. This configuration is particularly efficient if the columns of A are not strongly correlated, otherwise the loss of information is too significant and it causes $\sigma'_{HC}, \sigma'_{C} >> 1$: consequently, even if iterations are really inexpensive, their number is prohibitively high even for a parallel framework.



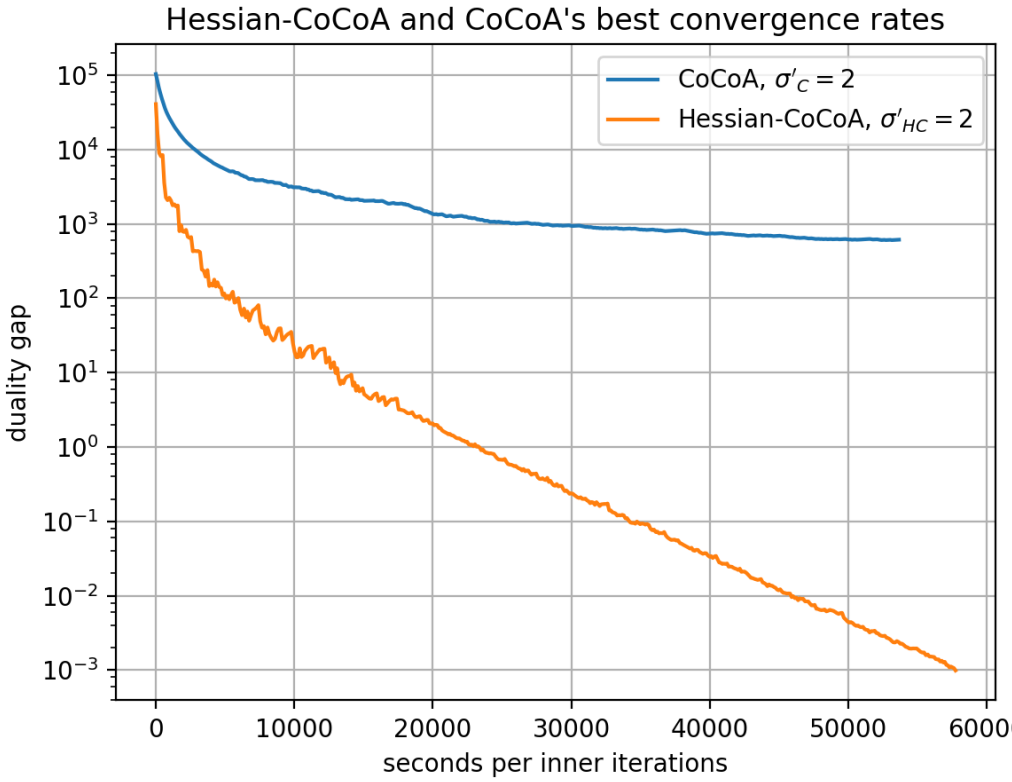
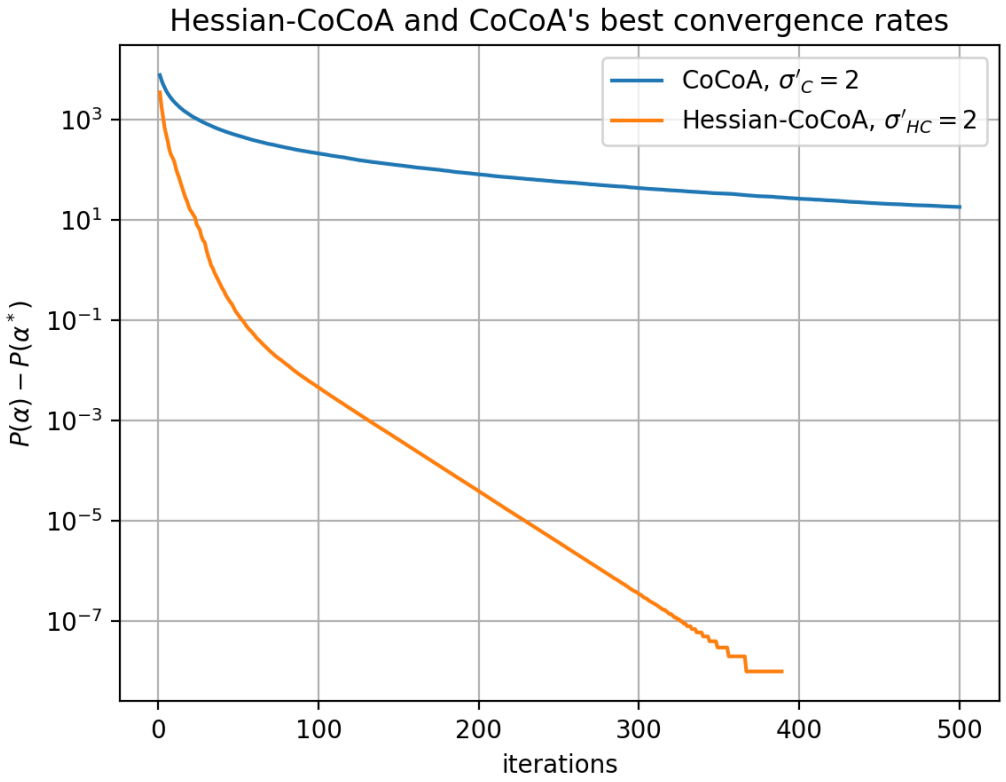


Figure 4.1: rcv1 data set, duality gap vs. number of iterations and seconds per inner iterations respectively, $n = 47.236$, $d = 20.242$, first configuration: $\bar{K} = K$, $K = 2$.



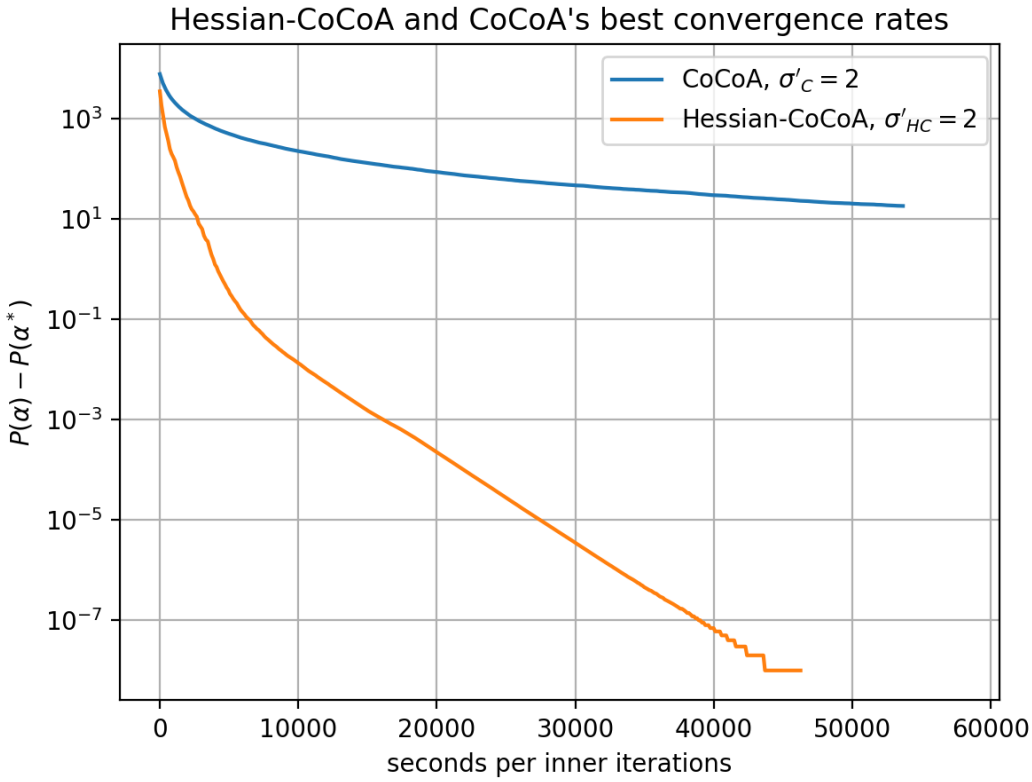
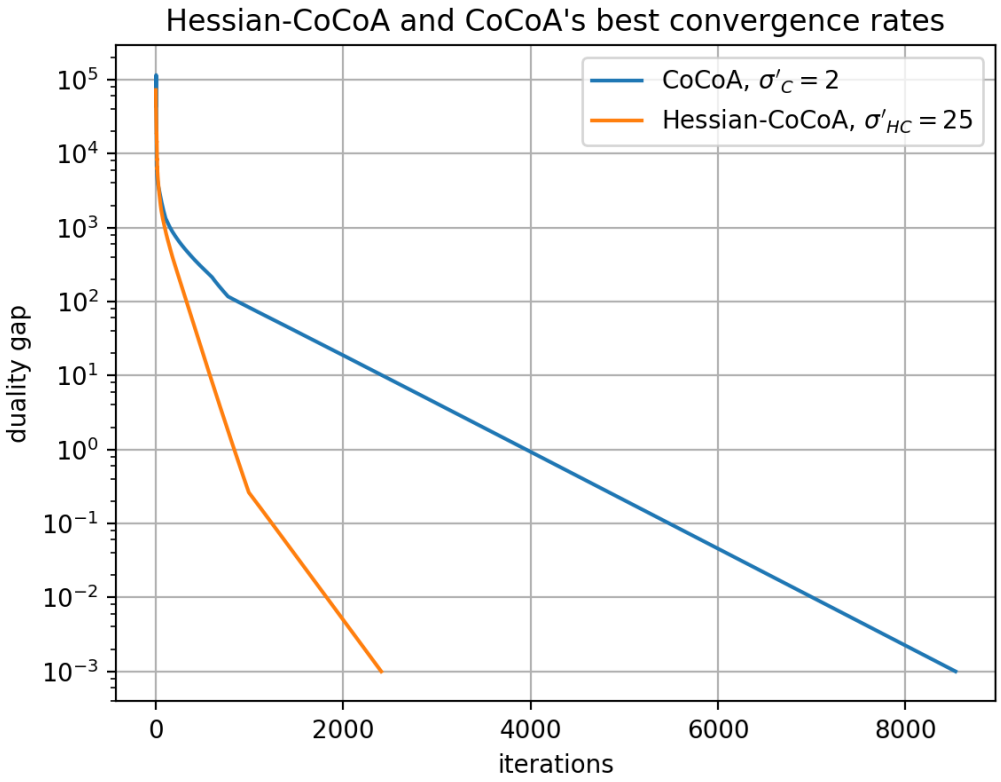


Figure 4.2: rcv1 data set, primal suboptimality vs. number of iterations and seconds per inner iterations respectively, $n = 47.236$, $d = 20.242$, first configuration: $\bar{K} = K$, $K = 2$.



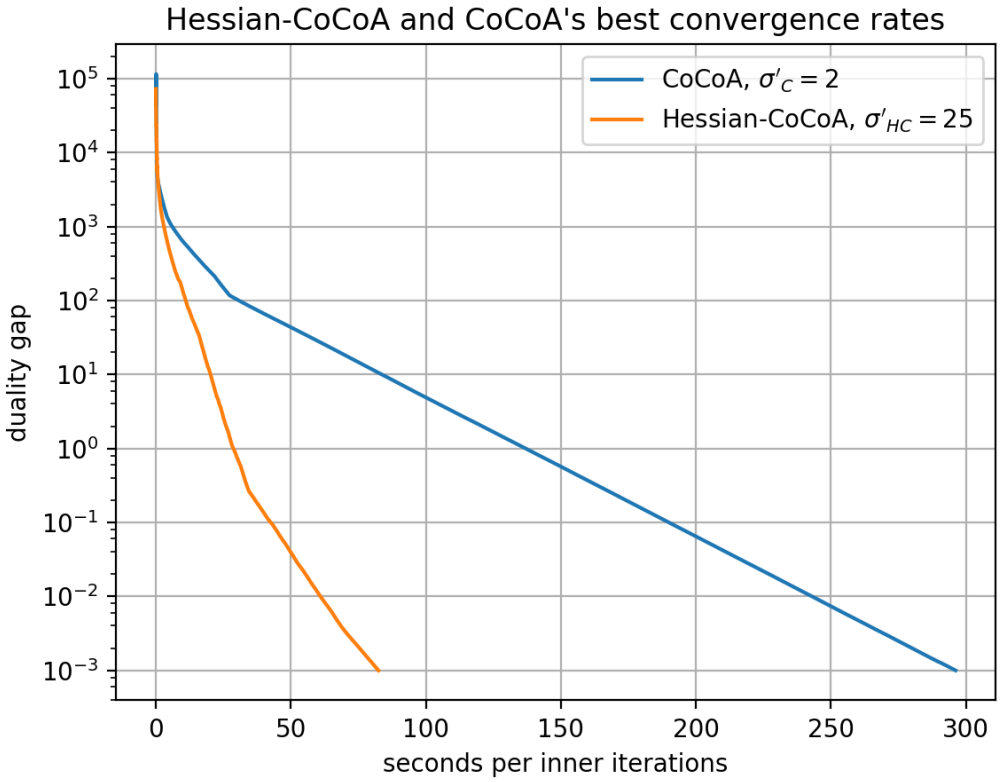
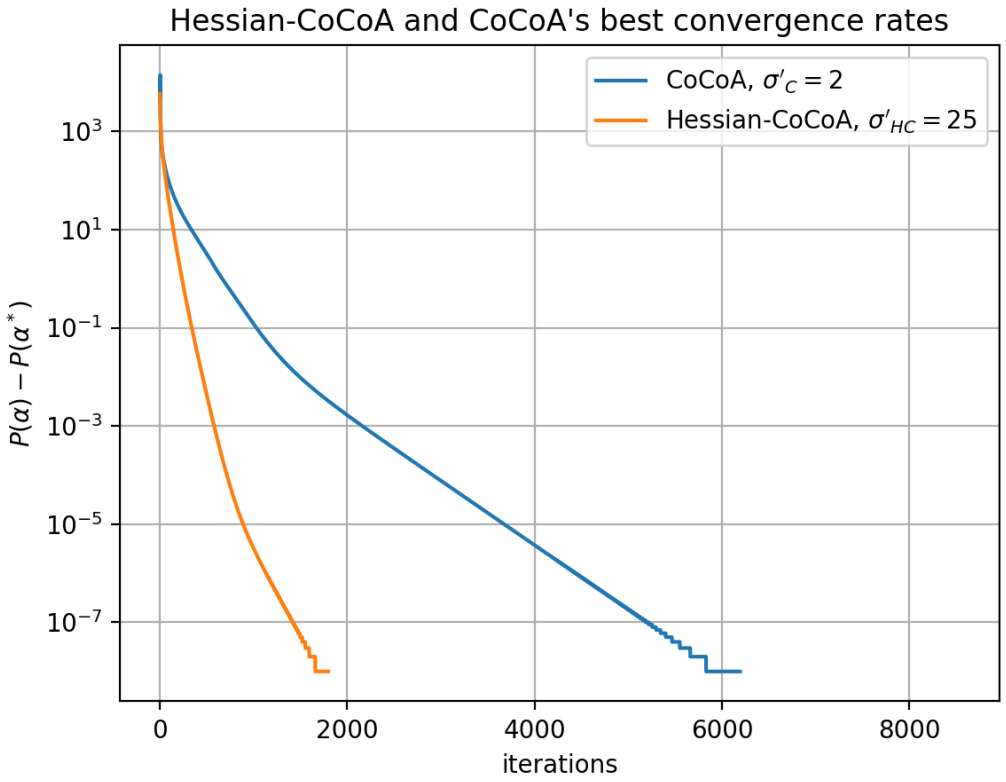


Figure 4.3: rcv1 data set, duality gap vs. number of iterations and seconds per inner iterations respectively, $n = 47.236$, $d = 20.242$, second configuration: $\bar{K} = n$, $K = 2$.



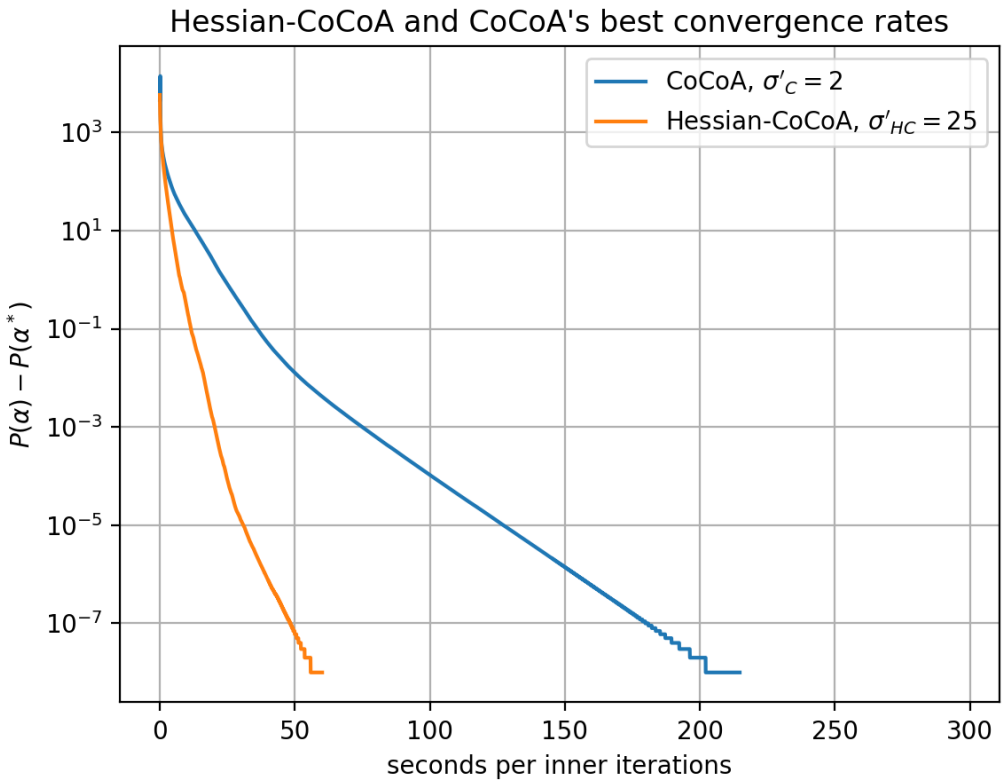
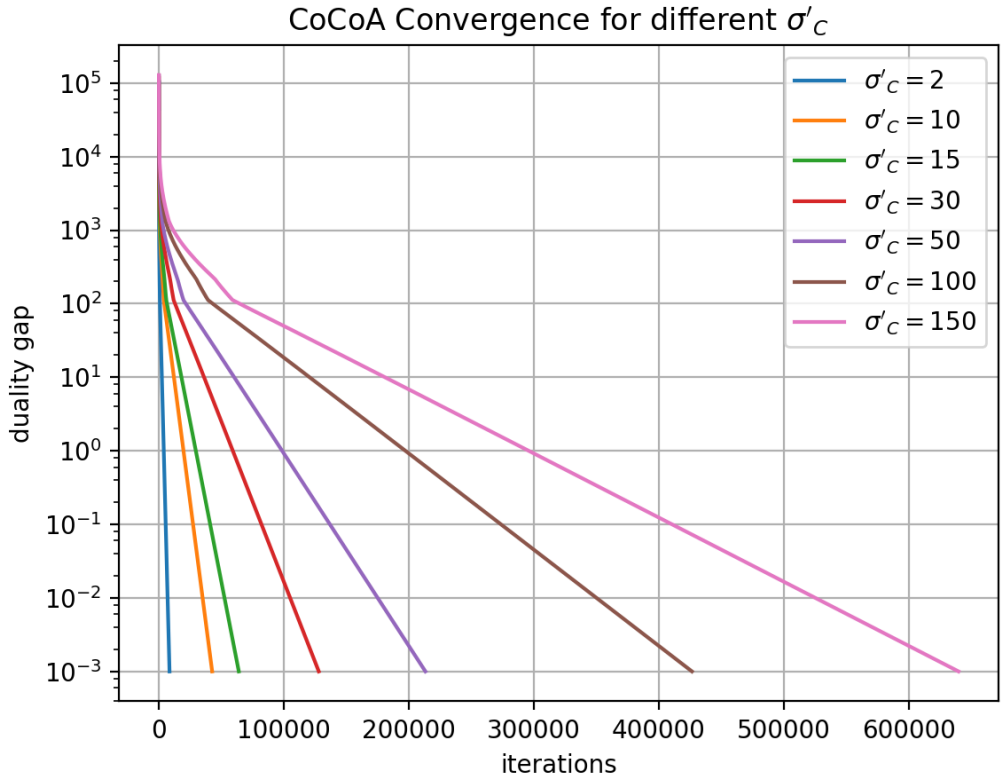


Figure 4.4: rcv1 data set, primal suboptimality vs. number of iterations and seconds per inner iterations respectively, $n = 47.236$, $d = 20.242$, second configuration: $\tilde{K} = n$, $K = 2$.



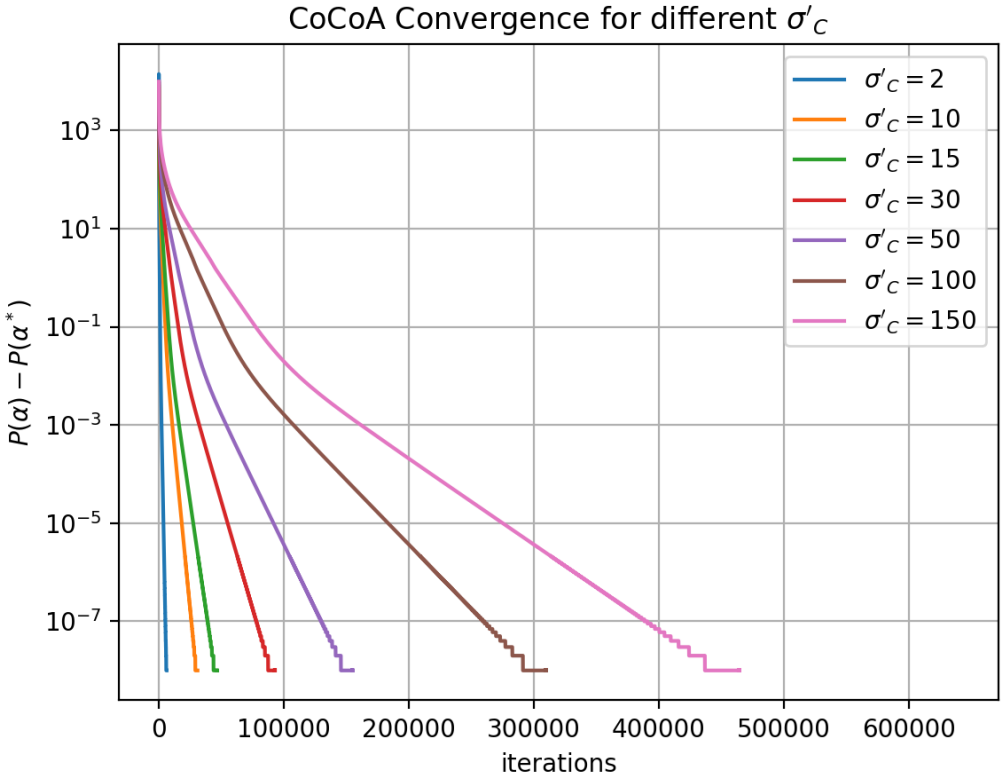
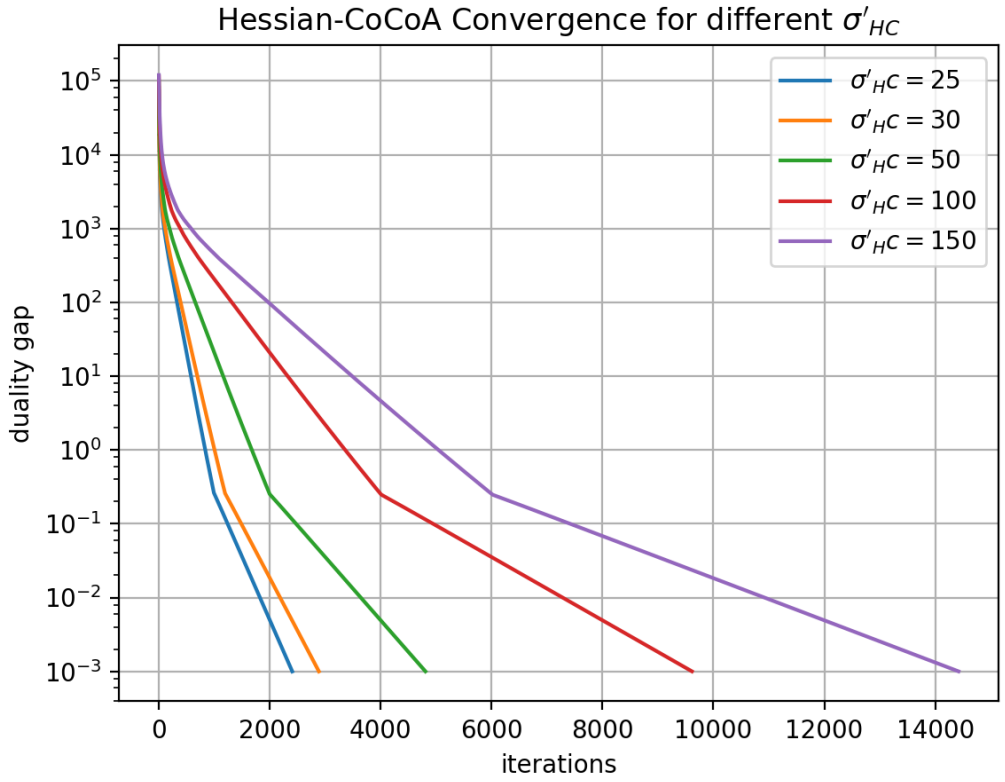


Figure 4.5: rcv1 data set, duality gap and primal suboptimality respectively vs. number of iterations, $n = 47.236$, $d = 20.242$, second configuration: $\bar{K} = n$.



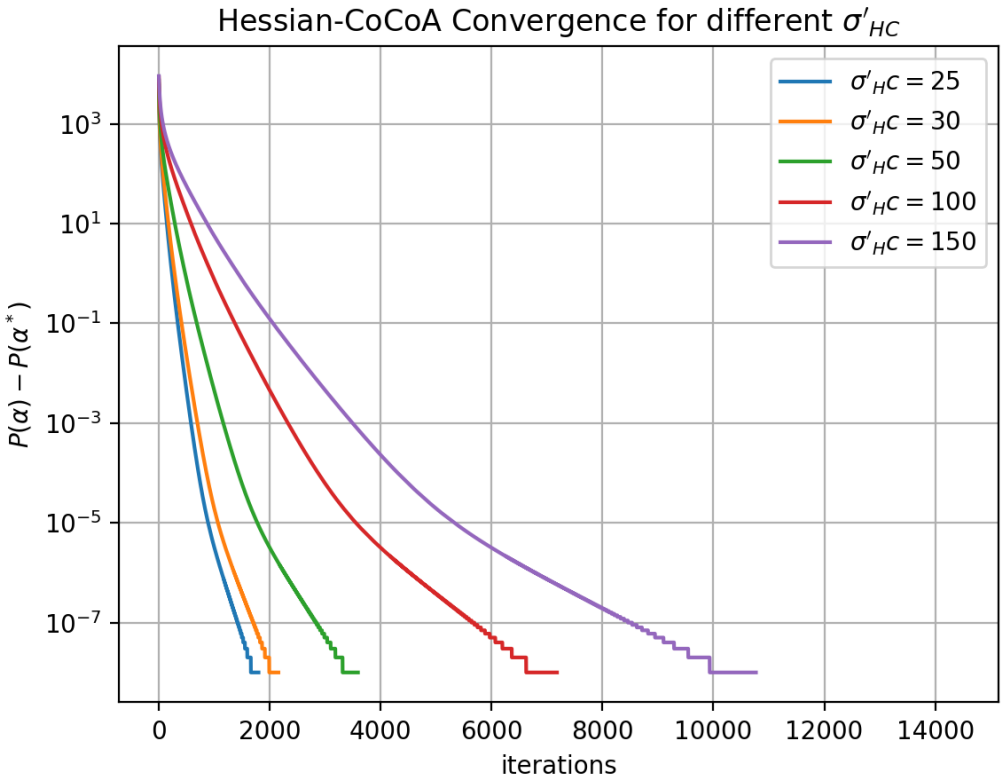
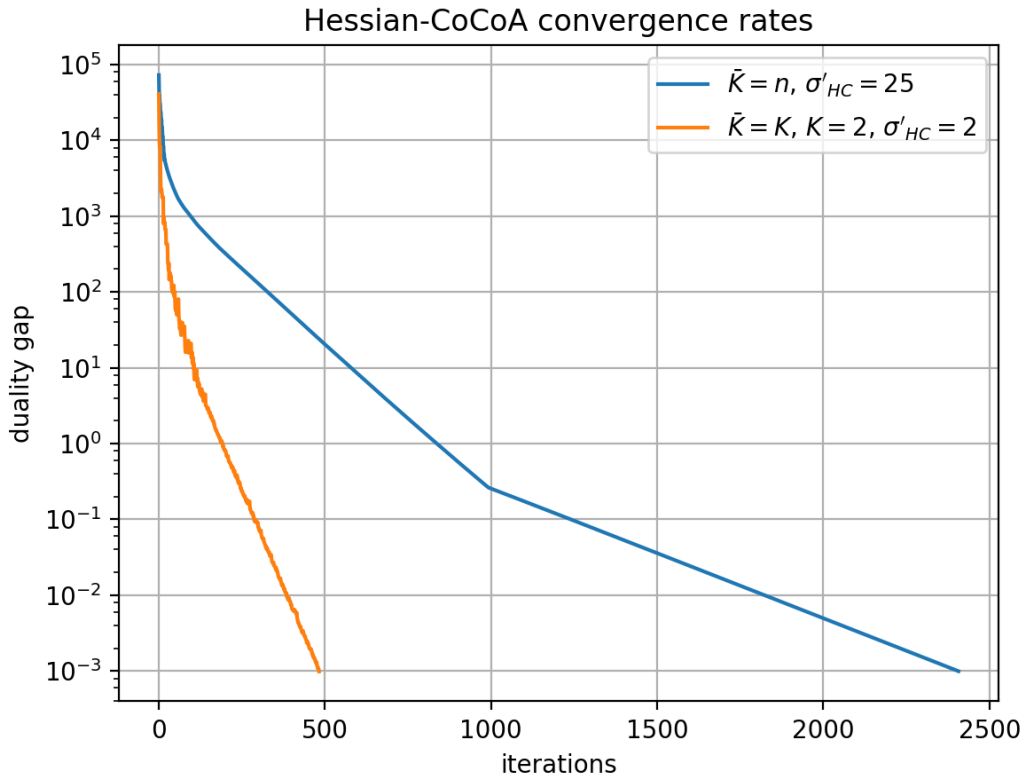


Figure 4.6: rcv1 data set, duality gap and primal suboptimality respectively vs. number of iterations, $n = 47.236$, $d = 20.242$, second configuration: $\bar{K} = n$.



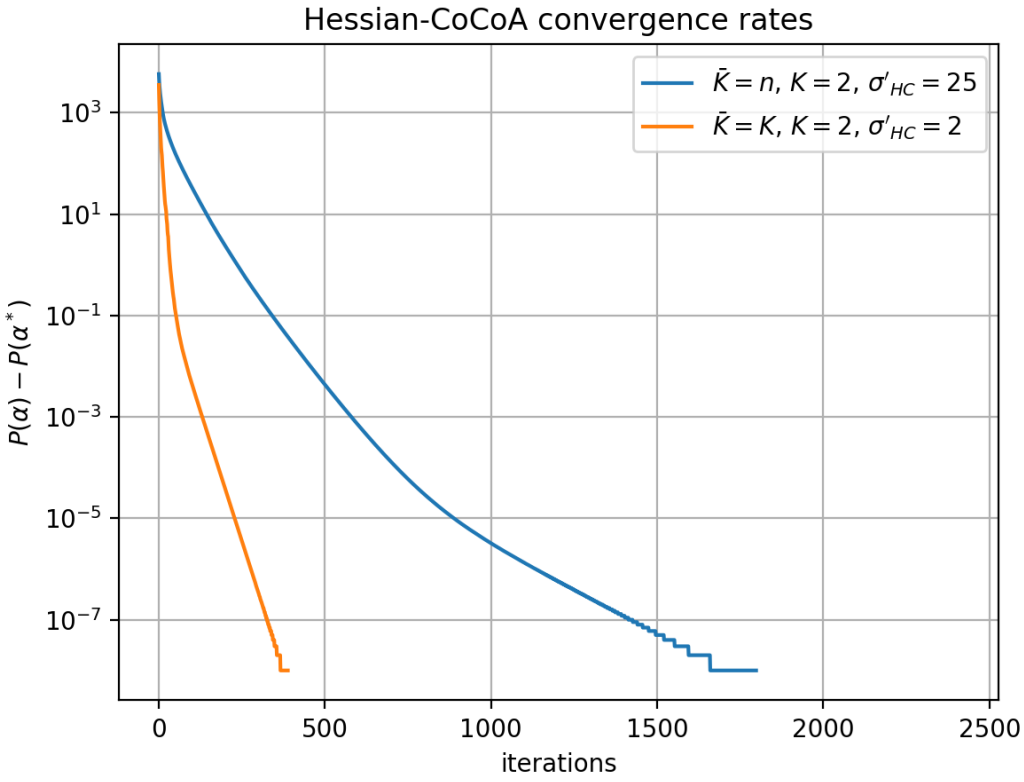


Figure 4.7: rcv1 data set, duality gap and primal suboptimality respectively vs. number of iterations for first and second configuration, $n = 47.236$, $d = 20.242$.

Chapter 5

Parallel Framework

Many applications uses ℓ_1-regularization, as Lasso and sparse Logistic Regression: ideed, because of its property of inducing sparsity in the solution, it is particularly useful for high-dimensional problems with a large number of features. Many sequential algorithms have been developed to solve this class of problems: from stochastic coordinate descent, together with its Newton version, to much more complex interior point methods (see the paper [6]). Despite the good performance of some sequential algorithms, high-dimensional data require scalable optimization methods: indeed, given the big and fast-growing dimensions of data sets, and, considering that processor core speeds have stopped increasing in recent years, many research projects have been done on how to efficiently use multicore machines. Therefore, a lot of efforts have been spent to exploit parallel computation and/or for completely parallelizing the existing sequential algorithms. Despite that, what the literature currently offers is not satisfying yet the requirements: focusing exclusively on parallel methods over features, the state-of-the-art is represented by Shotgun Coordinate Descent Newton, described in [5], [15], that originates from the ineherently sequential stochastic Coordinate Descent Newton (CDN) method. After a brief overview over stochastic CDN and its naive parallel version Shotgun-CDN, in this chapter we extend the analysis of Hessian-CoCoA to the parallel setting: we now assume the data matrix to be stored in a single memory shared across K processors working in parallel, and we introduce the degree of parallelization P, meaning the number of variables updated independently in each iteration by the K processors. The runtime guarantee that we have derived for parallel Hessian-CoCoA clearly shows that our algorithm is competitive with the state-of-the-art, since, by considering a specific setup, it is equivalent to Shotgun-CDN for the cases where this last is applicable, but it is also able to ensure convergence where Shotgun-CDN is not, with a rate that depedends on the correlation between features.

5.1 State-of-the-Art

In the following section, we analyze the state-of-the-art algorithm for solving ERM problems with an ℓ_1-regularizer in parallel over the number of features: Shotgun-CDN. As already mentioned, it has been originated from stochastic CDN, by naively parallelizing it. Despite it currently represents the state-of-the-art, it shows a big limitation: indeed, Shotgun-CDN does not allow for an arbitrary degree of parallelization, but the number of parallel updates is strongly constrained by the data matrix structure, and, in particular, by the correlation among the features. Starting from Shotgun-CDN, other methods have been proposed: for instance, Parallel-CDN, described by Bian et al. [15], whose convergence is ensured for any possible degree of parallelization, but, at the same time, in order to ensure that, it requires in each iteration a multidimensional line search over the set of the parallelized variables. Consequently, the bigger is the parallelization degree, the more computationally expensive is each iteration. Based on these considerations, ideally, we would like to develop a parallel algorithm with a competitive convergence rate, but, at the same time, that allows for any arbitrary choice of parallelization degree, without worsening the time cost per iteration.

5.1.1 Stochastic CDN

Stochastic Coordinate Descent Newton is an ineherently sequential algorithm that has been originated from Stochastic Coordinate Gradient Descent by partially including second order information. In particular, in iteration t, a feature $i \in \{1, \dots, n\}$ is selected uniformly at random, and then the direction $\Delta\alpha_i$ is computed by solving the optimization problem

$$\Delta\alpha_i = \arg\min_{\Delta\alpha \in \mathbb{R}} \left\{ (\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i \Delta\alpha + \frac{1}{2}\Delta\alpha^\top (A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)})A)_{[i,i]}\Delta\alpha + \lambda|\alpha_i^{(t)} + \Delta\alpha| \right\},$$

which has the following closed form solution

$$\Delta\alpha_i = \begin{cases} -\frac{(\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i + \lambda}{A_i^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A_i} & if \;\; (\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i + \lambda \leq (A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)})A)_{[i,i]}\alpha_i^{(t)} \\ -\frac{(\nabla^2 f(A\boldsymbol{\alpha}^{(t)})^\top A)_i - \lambda}{A_i^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A_i} & if \;\; (\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i - \lambda \geq (A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)})A)_{[i,i]}\alpha_i^{(t)} \\ -\alpha_i & otherwise\,. \end{cases}$$

Initialization

The algorithm initializes the variables to zero: $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$.

Convergence Analysis

Shalev-Shwartz and Tewari have analyzed this sequential method (see [11] for more details), and have come out with the best known convergence bound for it.

Theorem 5.1 *Let $\boldsymbol{\alpha}^\star$ be a minimizer of $\mathcal{O}_A(\boldsymbol{\alpha}) = f(A\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha}\|_1$, where f is $(\frac{1}{\tau})$-smooth and twice differentiable, and let $\boldsymbol{\alpha}^{(T)}$ be the optimization variable at the end of iteration T of stochastic CDN. Then, as it has been proved in [11], we have that*

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(T)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] \leq \frac{n(\frac{1}{\tau}\|\boldsymbol{\alpha}^\star\|_2^2 + \mathcal{O}_A(\mathbf{0}))}{2(T+1)},$$

where n is the number of features.

5.1.2 Shotgun-CDN

Shotgun-CDN has been ideated by Bradley et al. [5], and it consists in a naive parallelization of the ineherently sequential stochastic CDN: the algorithm first determines the number of parallel updates P (degree of parallelism) based on the data and on the number of available processors K. Then, in each iteration, it uniformly picks P features, which are then updated indepenedenty by the K processors working in parallel. In each iteration, each processor computes the exact same minimization problem as one iteration of stochastic CDN, with i being one of the variable assigned to the processor. In particular, given processor k, we indicate with $\mathcal{P}_k$ the subset of P variables that are updated by that processor, where $|\sum_{k=1}^{K} \mathcal{P}_k| = P$. Given iteration t, this globally results in the following optimization problem

$$\Delta\boldsymbol{\alpha} = \underset{\Delta\boldsymbol{\alpha}\in\mathbb{R}^P}{\arg\min}\left\{\sum_{i=1}^{P}(\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i\Delta\alpha_i + \frac{1}{2}\Delta\alpha^\top H_t\Delta\alpha + \lambda\sum_{i=1}^{P}|\alpha_i^{(t)} + \Delta\alpha_i|\right\}, \quad (5.1)$$

where $(H_t)_{[i,i]} = (A^\top\nabla^2 f(A\boldsymbol{\alpha}^{(t)})A)_{[i,i]}$, $\forall i = \{1,\dots,P\}$.

Initialization

The algorithm initializes the variables to zero: $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$.

Algorithm 4 Shotgun-CDN

Input: data matrix $A \in \mathbb{R}^{d \times n}$ stored in a single memory
choose the number of parallel updates $P \geq 1$
set $t = -1$, $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{v}^{(0)} := \mathbf{0} \in \mathbb{R}^d$
while not converged **do**
  $t = t + 1$
  choose P variables uniformly at random
  for $k \in \{1, 2, \ldots, K\}$ **in parallel over processors do**
    for $i \in \mathcal{P}_k$ **do**
      obtain $\Delta\alpha_i$ by solving the i component of the global problem (5.1)
      update $\alpha_i^{(t+1)} = \alpha_i^{(t)} + \Delta\alpha_i$
    end for
    wait all processors to finish the update
  end for
  $\mathbf{v}^{(t+1)} = A\boldsymbol{\alpha}^{(t+1)}$
end while

Here we consider a fixed and unitary step length version for both Shotgun and Shotgun-CDN, but already Bradley et al. in [5] had implemented a modified version by introducing line searches.

Convergence Analysis

Theorem 5.2 *Let $\boldsymbol{\alpha}^\star$ be the minimizer of $\mathcal{O}_A(\boldsymbol{\alpha}) = f(A\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha}\|_1$, where f is $\frac{1}{\tau}$- smooth and twice differentiable; and let $\boldsymbol{\alpha}^{(T)}$ the output of Shotgun-CDN after T iterations. Let ρ be the spectral radius of $A^\top A$, and P such that $\varepsilon = \frac{(P-1)(\rho-1)}{2n-1} < 1$, then*

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(T)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] \leq \frac{n\left(\frac{1}{\tau}\|\boldsymbol{\alpha}^\star\|_2^2 + \frac{2}{1-\varepsilon}\mathcal{O}_A(\mathbf{0})\right)}{(T+1)P},$$

where $\mathbb{E}[\cdot]$ is w.r.t. the random choices of variables to update in each iteration.

Shotgun-CDN shows a big limitation: in order to achieve convergence, $P \leq \frac{n}{\rho+1}$, where ρ is the spectral radius of $A^\top A$. Therefore, the allowed degree of parallelization P strongly depends on the application: the more the features are correlated to each others, the smaller the maximum possible P will result. Unfortunatley, this limitation is often too restrictive to enable parallelization (e.g. $\rho = 20228800$ for **gisette** dataset with $n = 5000$).

5.2 Parallel Diagonal Hessian-CoCoA

A first naive parallel version of Hessian-CoCoA consists in selecting P features uniformly at random, and parallelize their optimization by splitting the computation across K processors. In particular, as Shotgun-CDN, we consider a diagonal structure for the local approximation of the Hessian matrix, such that the one-variable local subproblems have a closed form solution that can be computed efficiently. This particular setup is chosen to enable a direct comparison with the state-of-the-art algorithms, but Hessian-CoCoA allows also for different configurations of the Hessian matrix. The strength of Hessian-CoCoA is that, thanks to the presence of σ'_{HC} parameter, convergence is ensured for any possible value of P, without the need of line search, and theoretically enabling also for $P = n$. Again, as for Shotgun-CDN, in each iteration P variables are selected uniformly at random, and then each of the K processors works in parallel over a subset of variables $\mathcal{P}_k$, where $|\sum_{k=1}^{K} \mathcal{P}_k| = P$.

Given iteration t, processor i computes the direction $\Delta\alpha_i$, where

$$\Delta\alpha_i = \arg\min_{\Delta\alpha\in\mathbb{R}} \left\{ (\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i \Delta\alpha + \frac{\sigma'_{HC}}{2} \Delta\alpha^\top (A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A)_{[i,i]} \Delta\alpha + \lambda |\alpha_i^{(t)} + \Delta\alpha| \right\}, \tag{5.2}$$

which globally results in the following optimization problem

$$\Delta\boldsymbol{\alpha} = \arg\min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^P} \left\{ \sum_{i=1}^{P} (\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i \Delta\alpha_i + \frac{\sigma'_{HC}}{2} \Delta\alpha^\top H_t \Delta\alpha + \lambda \sum_{i=1}^{P} |\alpha_i^{(t)} + \Delta\alpha_i| \right\},$$

where $(H_t)_{[i,i]} = (A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A)_{[i,i]}$, $\forall i \in \{1, \ldots, P\}$.

Initialization

The algorithm initializes the variables to zero: $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$.

Algorithm 5 Parallel Diagonal Hessian-CoCoA

Input: data matrix $A \in \mathbb{R}^{d \times n}$ stored in a single memory, step length $\gamma \in (0, 1]$, and parameter σ'_{HC} for the local subproblems
set $t = -1$, $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{v}^{(0)} := \mathbf{0} \in \mathbb{R}^d$
while not converged **do**
  $t = t + 1$
  for $k \in \{1, 2, \ldots, K\}$ **in parallel over processors do**
    for $i \in \mathcal{P}_k$ **do**
      obtain $\Delta\alpha_i$ by solving the i component of the global problem (5.2)
      update $\alpha_i^{(t+1)} = \alpha_i^{(t)} + \gamma\Delta\alpha_i$
    end for
    wait all processors to finish the update
  end for
  $\mathbf{v}^{(t+1)} = A\boldsymbol{\alpha}^{(t+1)}$
end while

5.3 Convergence Analysis

For simplicity, the convergence analysis has been conducted for the CoCoA case by setting $\gamma = 1$, and it can be generally considered valid also for Hessian-CoCoA, since, by setting σ'_{HC} and σ'_C to an equal safe value, it has been shown that Hessian-CoCoA converges faster since it is based on a local tighter model (see section 3.3.3 and section 5.3.3).

Normalization

The data matrix $A \in \mathbb{R}^{d \times n}$ is normalized such that $diag(A^\top A) = I$.

5.3.1 σ'_C Parameter

Let $A_{[P]} \in \mathbb{R}^{d \times P}$ be the matrix obtained by selecting from $A \in \mathbb{R}^{d \times n}$ the P columns corresponding to the set of selected variables $\mathcal{P}_t$. Recalling that we are now working with a diagonal approximation for the local Hessian matrix, the resulting σ'_C parameter is

$$\sigma'_C \geq \max_{\substack{\Delta\boldsymbol{\alpha} \neq \mathbf{0} \\ \Delta\boldsymbol{\alpha} \in \mathbb{R}^P}} \frac{\Delta\boldsymbol{\alpha}^\top A_{[P]}^\top A_{[P]} \Delta\boldsymbol{\alpha}}{\Delta\boldsymbol{\alpha}^\top \Delta\boldsymbol{\alpha}} := \rho(A_{[P]}^\top A_{[P]}) \quad \forall \ \mathcal{P}_t\,, \ |\mathcal{P}_t| = P\,.$$

In case that $P = n$, then

$$\sigma'_C \geq \max_{\substack{\Delta\boldsymbol{\alpha} \neq \mathbf{0} \\ \Delta\boldsymbol{\alpha} \in \mathbb{R}^n}} \frac{\Delta\boldsymbol{\alpha}^\top A^\top A \Delta\boldsymbol{\alpha}}{\Delta\boldsymbol{\alpha}^\top \Delta\boldsymbol{\alpha}} := \rho(A^\top A)\,.$$

5.3.2 σ'_{HC} Parameter

Regarding σ'_{HC}, we could drive similar conclusions: indeed it is easy to see from its definition in equation (3.7) that its value is related to the features correlation. In particular

$$\sigma'_{HC} \propto \max_{\substack{\mathcal{P}_t, \\ |\mathcal{P}_t|=P}} \max_{\theta\in[0,1]} \rho(A_{[P]}^\top \nabla^2 f(A(\boldsymbol{\alpha}+\theta\Delta\boldsymbol{\alpha}))A_{[P]}) \leq \frac{1}{\tau} \max_{\substack{\mathcal{P}_t, \\ |\mathcal{P}_t|=P}} \rho(A_{[P]}^\top A_{[P]}),$$

and, in case of a full degree of parallelization, i.e. $P = n$, we have

$$\sigma'_{HC} \propto \rho(A^\top A) \leq \frac{1}{\tau}\rho(A^\top A).$$

5.3.3 Runtime Guarantee

The following inequality is foundamental to consider the parallel CoCoA's runtime guarantee valid also for the parallel Hessian-CoCoA case.

$$\begin{aligned}
\mathcal{O}_A(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}) &\leq f(A\boldsymbol{\alpha}) + (\nabla f(A\boldsymbol{\alpha}))^\top A_{[P]}\Delta\boldsymbol{\alpha}_{[P]} + \frac{\sigma'_{HC}}{2}\Delta\boldsymbol{\alpha}_{[P]}^\top A_{[P]}^\top \nabla^2 f(A\boldsymbol{\alpha})A_{[P]}\Delta\boldsymbol{\alpha}_{[P]} \\
&\quad + \sum_{i\in\mathcal{P}_t} g_i(\boldsymbol{\alpha}_i+\Delta\boldsymbol{\alpha}_i) + \sum_{i\notin\mathcal{P}_t} g_i(\boldsymbol{\alpha}_i) \\
&\leq f(A\boldsymbol{\alpha}) + (\nabla f(A\boldsymbol{\alpha}))^\top A_{[P]}\Delta\boldsymbol{\alpha}_{[P]} + \frac{\sigma'_{HC}}{2\tau}\Delta\boldsymbol{\alpha}_{[P]}^\top\Delta\boldsymbol{\alpha}_{[P]} \\
&\quad + \sum_{i\in\mathcal{P}_t} g_i(\boldsymbol{\alpha}_i+\Delta\boldsymbol{\alpha}_i) + \sum_{i\notin\mathcal{P}_t} g_i(\boldsymbol{\alpha}_i).
\end{aligned}$$

Theorem 5.3 *Let $\boldsymbol{\alpha}^\star$ be the minimizer of $\mathcal{O}_A(\boldsymbol{\alpha}) = f(A\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha}\|_1$, where f is $\frac{1}{\tau}$- smooth and twice differentiable. Let $\boldsymbol{\alpha}^{(T)}$ be the output of parallel CoCoA after T iterations. In addition, let σ'_C be in the safe range of values for the given degree of parallelization P. Then*

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(T)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] \leq \frac{n\left(\frac{\sigma'_C}{\tau}\|\boldsymbol{\alpha}^\star\|_2^2 + 2\mathcal{O}_A(\mathbf{0})\right)}{2(T+1)P},$$

where $\mathbb{E}[\cdot]$ is w.r.t. the random choices of variables to update in each iteration.

Proof As first step, we analyze $\Delta\alpha_i$, where $\Delta\alpha_i = S_{\frac{\lambda}{\sigma'_C\frac{1}{\tau}}}(\alpha_i - \frac{(\nabla f(A\boldsymbol{\alpha})^\top A)_i}{\sigma'_C\frac{1}{\tau}}) - \alpha_i$.
This is equivalent to claim that

$$\Delta\alpha_i = \arg\min_{\Delta\alpha\in\mathbb{R}} \left\{ (\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i \Delta\alpha + \frac{\sigma_C'}{2\tau}(\Delta\alpha)^2 + \lambda|\alpha_i^{(t-1)} + \Delta\alpha| \right\} .$$

Indeed, the optimality condition for the above optimization problem is

$$0 = (\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i + \frac{\sigma_C'}{\tau}\Delta\alpha + \lambda\nu_i \,,$$

where $\nu_i \in \delta|\alpha_i^{(t-1)} + \Delta\alpha|$ is the subdifferential of the absolute value at $\alpha_i^{(t-1)} + \Delta\alpha$. Given that $\nu_i = sign(\alpha_i^{(t-1)} + \Delta\alpha)$ if $\alpha_i^{(t-1)} + \Delta\alpha \neq 0$, and $\nu_i \in [-1, 1]$ otherwise, we have the following three cases:

- $\Delta\alpha > -\alpha_i^{(t-1)} \implies \nu_i = 1 \implies \Delta\alpha = \frac{-(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i - \lambda}{\frac{\sigma_C'}{\tau}} > -\alpha_i^{(t-1)}$,
- $\Delta\alpha < -\alpha_i^{(t-1)} \implies \nu_i = -1 \implies \Delta\alpha = \frac{-(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i + \lambda}{\frac{\sigma_C'}{\tau}} < -\alpha_i^{(t-1)}$,
- $\Delta\alpha = -\alpha_i^{(t-1)}$.

But this is nothing but the definition of $\Delta\alpha_i$.

We then proceed by defining the potential function

$$\Phi(\boldsymbol{\alpha}^{(t)}) = \frac{1}{2}\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\|_2^2 \,, \tag{5.3}$$

and by bounding $\Delta_{t,\mathcal{P}_t} = \Phi(\boldsymbol{\alpha}^{(t-1)}) - \Phi(\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha})$, where $\Delta\alpha_i = 0 \; \forall i \notin \mathcal{P}_t$

$$\begin{aligned}
&\Delta_{t,\mathcal{P}_t} := \Phi(\boldsymbol{\alpha}^{(t-1)}) - \Phi(\boldsymbol{\alpha}^{(t)}) = \frac{1}{2}\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t-1)}\|_2^2 - \frac{1}{2}\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t-1)} - \Delta\boldsymbol{\alpha}\|_2^2 \\
&= \sum_{i\in\mathcal{P}_t} \left[\frac{1}{2}(\alpha_i^\star - \alpha_i^{(t-1)})^2 - \frac{1}{2}(\alpha_i^\star - \alpha_i^{(t-1)} - \Delta\alpha_i)^2 \right] \\
&= \sum_{i\in\mathcal{P}_t} \left[\frac{1}{2}(\Delta\alpha_i)^2 + \frac{(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i}{\frac{\sigma_C'}{\tau}}(\alpha_i^{(t-1)} + \Delta\alpha_i - \alpha_i^\star) + \frac{\lambda\nu_i}{\frac{\sigma_C'}{\tau}}(\alpha_i^{(t-1)} + \Delta\alpha_i - \alpha_i^\star) \right] .
\end{aligned}$$

Since ν_i is a subdifferential, it follows that

$$\nu_i(\alpha_i^{(t-1)} + \Delta\alpha_i - \alpha_i^\star) \leq |\alpha_i^{(t-1)} + \Delta\alpha_i| - |\alpha_i^\star| \,.$$

We also have that

$$f(A\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha}) - f(A\boldsymbol{\alpha}^{(t-1)}) \leq (\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)\Delta\boldsymbol{\alpha} + \frac{\sigma_C'}{2\tau}(\Delta\boldsymbol{\alpha})^\top(\Delta\boldsymbol{\alpha}) \,.$$

Consequently,

$$\Delta_{t,\mathcal{P}_t} \geq \frac{\tau}{\sigma'_C}\left(f(A\boldsymbol{\alpha}^{(t-1)}+\Delta\boldsymbol{\alpha})-f(A\boldsymbol{\alpha}^{(t-1)})\right)+ \\ +\sum_{i\in\mathcal{P}_t}\frac{(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i}{\frac{\sigma'_C}{\tau}}(\alpha_i^{(t-1)}-\alpha_i^\star)+\frac{\lambda}{\frac{\sigma'_C}{\tau}}(|\alpha_i^{(t-1)}+\Delta\alpha_i|-|\alpha_i^\star|)\,.$$

Taking the expectation value, conditioned of $\boldsymbol{\alpha}^{(t-1)}$, we obtain that

$$\mathbb{E}_{\mathcal{P}_t}\left[\Phi(\boldsymbol{\alpha}^{(t-1)})-\Phi(\boldsymbol{\alpha}^{(t)})\right]=\mathbb{E}_{\mathcal{P}_t}\left[\sum_{i\in\mathcal{P}_t}\Delta_{t,i}\right]=P\,\mathbb{E}_i\left[\Delta_{t,i}\right]$$

$$\geq\frac{P\tau}{n\sigma'_C}\left[\sum_{i=1}^n\left(f(A\boldsymbol{\alpha}^{(t-1)}+\Delta\boldsymbol{\alpha}\mathbf{e}_i)-f(A\boldsymbol{\alpha}^{(t-1)})\right)\right]+\frac{P\tau}{n\sigma'_C}\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A(\boldsymbol{\alpha}^{(t-1)}-\boldsymbol{\alpha}^\star)+$$

$$+\frac{P\tau}{n\sigma'_C}\lambda\sum_{i=1}^n\left(|\alpha_i^{(t-1)}+\Delta\alpha_i|-|\alpha_i^\star|\right)$$

$$\geq\frac{P\tau}{n\sigma'_C}\left[\sum_{i=1}^n\left(f(A\boldsymbol{\alpha}^{(t-1)}+\Delta\boldsymbol{\alpha}\mathbf{e}_i)-f(A\boldsymbol{\alpha}^{(t-1)})\right)\right]+\frac{P\tau}{n\sigma'_C}\left[f(A\boldsymbol{\alpha}^{(t-1)})-f(A\boldsymbol{\alpha}^\star)\right]+$$

$$+\frac{P\tau}{n\sigma'_C}\lambda\sum_{i=1}^n\left(|\alpha_i^{(t-1)}+\Delta\alpha_i|-|\alpha_i^\star|\right)$$

$$=\frac{\tau}{\sigma'_C}\left[\mathbb{E}\left[f(A\boldsymbol{\alpha}^{(t)})-f(A\boldsymbol{\alpha}^{(t-1)})\right]+\frac{P\left(f(A\boldsymbol{\alpha}^{(t-1)})-f(A\boldsymbol{\alpha}^\star)\right)}{n}\right]+$$

$$+\frac{\tau}{\sigma'_C}\left[\frac{P\lambda}{n}\sum_{i=1}^n|\alpha_i^{(t-1)}+\Delta\alpha_i|-\lambda P\frac{\|\boldsymbol{\alpha}^\star\|}{n}\right]$$

Note that, we have

$$\mathbb{E}\left[\|\boldsymbol{\alpha}^{(t)}\|_1\right]=\frac{P}{n}\sum_{i=1}^n\|\boldsymbol{\alpha}^{(t-1)}+\Delta\boldsymbol{\alpha}\mathbf{e}_i\|_1 \\ =\|\boldsymbol{\alpha}^{(t-1)}\|_1-\frac{P}{n}\|\boldsymbol{\alpha}^{(t-1)}\|_1+\frac{P}{n}\sum_{i=1}^n|\boldsymbol{\alpha}_i^{(t-1)}+\Delta\alpha_i|\,.$$

Plugging this in the above equation, gives us

$$\frac{\sigma'_C}{\tau}\mathbb{E}\left[\Phi(\boldsymbol{\alpha}^{(t-1)})-\Phi(\boldsymbol{\alpha}^{(t)})\right] \\ \geq\mathbb{E}\left[f(A\boldsymbol{\alpha}^{(t)})+\lambda\|\boldsymbol{\alpha}^{(t)}\|_1-f(A\boldsymbol{\alpha}^{(t-1)})-\lambda\|\boldsymbol{\alpha}^{(t-1)}\|_1\right]+ \\ +\frac{P\left(f(A\boldsymbol{\alpha}^{(t-1)})+\lambda\|\boldsymbol{\alpha}^{(t-1)}\|_1-f(A\boldsymbol{\alpha}^\star)-\lambda\|\boldsymbol{\alpha}^\star\|_1\right)}{n} \\ =\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)})-\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)})\right]+\frac{P(\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)})-\mathcal{O}_A(\boldsymbol{\alpha}^\star))}{n}\,.$$

This is equivalent to

$$\mathbb{E}\left[\frac{\sigma'_C}{\tau}\Phi(\boldsymbol{\alpha}^{(t-1)}) + \mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \frac{\sigma'_C}{\tau}\Phi(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t)})\right] \geq \frac{P(\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star))}{n}\,.$$

We now define the composite potential function

$$\Psi(\boldsymbol{\alpha}) = \frac{\sigma'_C}{\tau}\Phi(\boldsymbol{\alpha}) + \mathcal{O}_A(\boldsymbol{\alpha})\,,$$

and, taking full expectation, we get

$$\mathbb{E}\left[\Psi(\boldsymbol{\alpha}^{(t-1)}) - \Psi(\boldsymbol{\alpha}^{(t)})\right] \geq \frac{P}{n}\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right]\,.$$

Summing over $t = 1, \ldots, T$ and realizing that $\mathcal{O}_A(\boldsymbol{\alpha}^{(t)})$ monotonically decreases, we obtain

$$\begin{aligned}\frac{P}{n}\mathbb{E}\left[(T+1)(\mathcal{O}_A(\boldsymbol{\alpha}^{(T)})) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] &\leq \frac{P}{n}\mathbb{E}\left[\sum_{t=1}^{T+1}(\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star))\right] \\ &\leq \mathbb{E}\left[\sum_{t=1}^{T+1}(\Psi(\boldsymbol{\alpha}^{(t-1)}) - \Psi(\boldsymbol{\alpha}^{(t)}))\right] \\ &= \mathbb{E}\left[(\Psi(\mathbf{0}) - \Psi(\boldsymbol{\alpha}^{T+1}))\right] \leq \mathbb{E}\left[\Psi(\mathbf{0})\right] = \Psi(\mathbf{0})\,.\end{aligned}$$

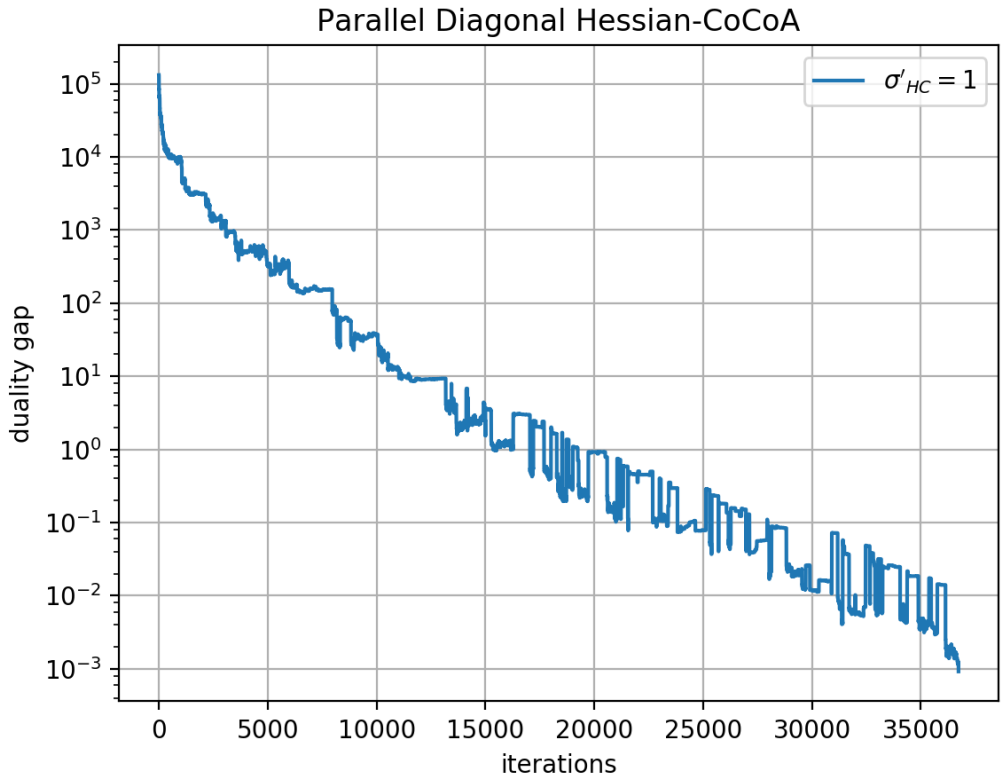
□

5.4 Experiments

Experiments have been conducted for ℓ_1-logistic regression case over two different datasets: **rcv1** and **duke** datasets. The parallel setting has been simulated on a single machine, in particular we have used a MacBook Pro with 2.7 GHz Intel Core i5 Processor, and, regarding the implementation, Python 2.7 with SciPy.sparse package to handle big and sparse matrices. For both the datasets, the degree of parallelization P has been set to 100, which corresponds to around the 0.2% and 1.4% of the total variables for the two cases respectively. For **rcv1** dataset, Hessian-CoCoA with a diagonal Hessian matrix configuration and Shotgun-CDN coincides, since $\sigma'_{HC} = 1$ is a safe value. Indeed, the only difference between the two methods consists in the parameter σ'_{HC} which accounts for the loss of information: for Shotgun-CDN it is implicitly present and always considered equal to 1, while in Hessian-CoCoA its value is tuned based on the data and the amount of information

lost. Despite the small value for the degree of parallelization, Shotgun-CDN clearly fails when applied to **duke** dataset: features are indeed significantly correlated and, therefore, the loss of information is preventing the method from achieving the convergence. While, thanks to the presence of σ'_{HC}, that it set to a safe value, Hessian-CoCoA converges, proving to be competitive with the state-of-the-art parallel algorithm.

Regarding the subproblems configuration, the choice of considering a diagonal Hessian martix is necessary for an easy comparison with Shotgun and Shotgun-CDN. At the same time, this configuration might not be the best one in terms of convergence rate and total convergence time. In particular, in the case of strongly correleted features, if we consider a diagonal Hessian matrix then Hessian-CoCoA might require a prohibitive high value for σ'_{HC} to ensure the convergence. For these datasets, in order to speed up the convergence, it might be appropriate to include more information in the local subproblems, by means of a block-diagonal structure for the Hessian matrix instead of just a diagonal one. Hessian-CoCoA differently from Shotgun and Shotgun-CDN, also allows for this kind of configuration, proving to be a more flexible parallel framework, able to adapt to different datasets.



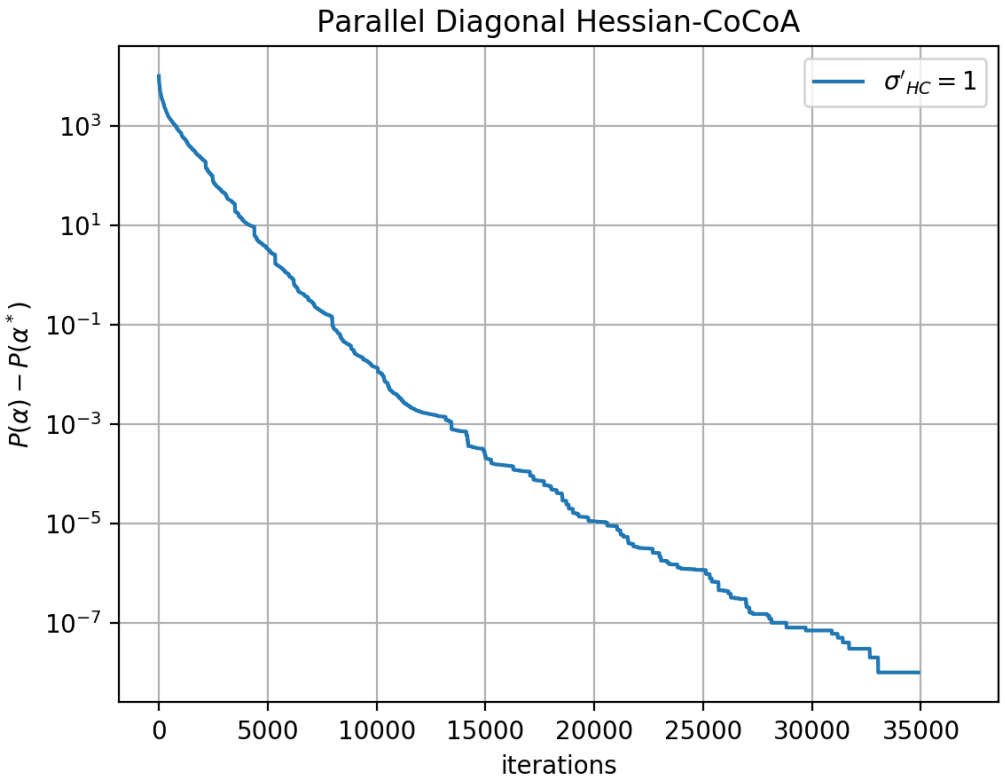
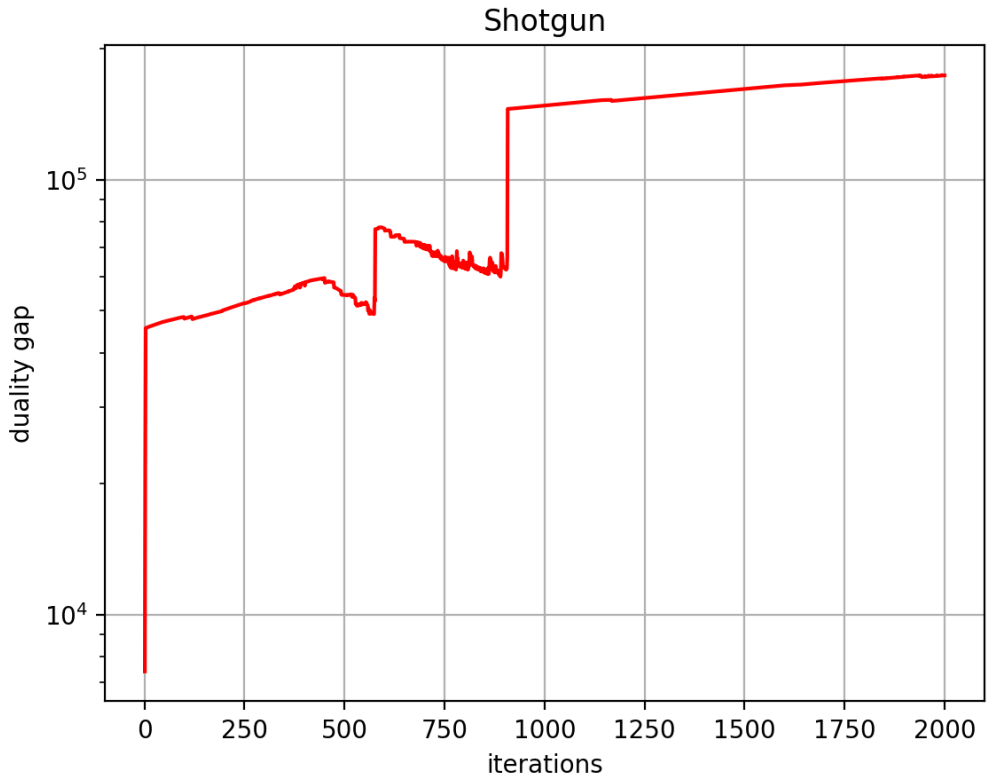


Figure 5.1: rcv1 data set, duality gap and primal suboptimality respectively vs. number of iterations, $n = 47.236$, $d = 20.242$, $P = 100$.



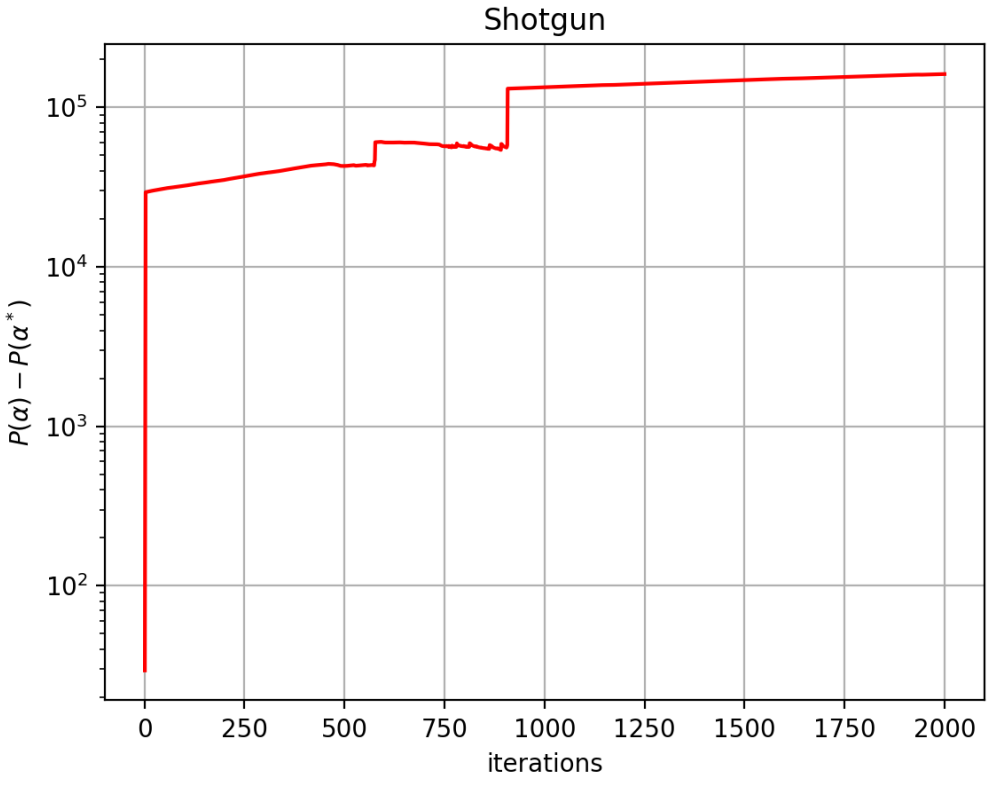
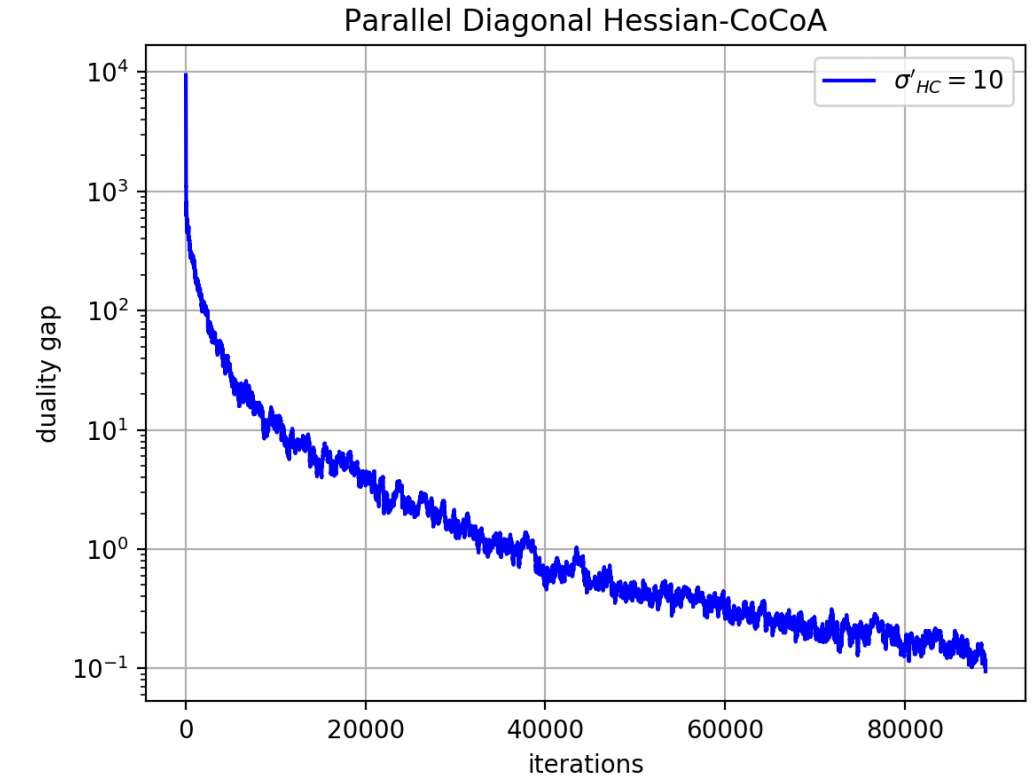


Figure 5.2: duke data set, duality gap and primal suboptimality respectively vs. number of iterations, $n = 7.129$, $d = 44$, $P = 100$.



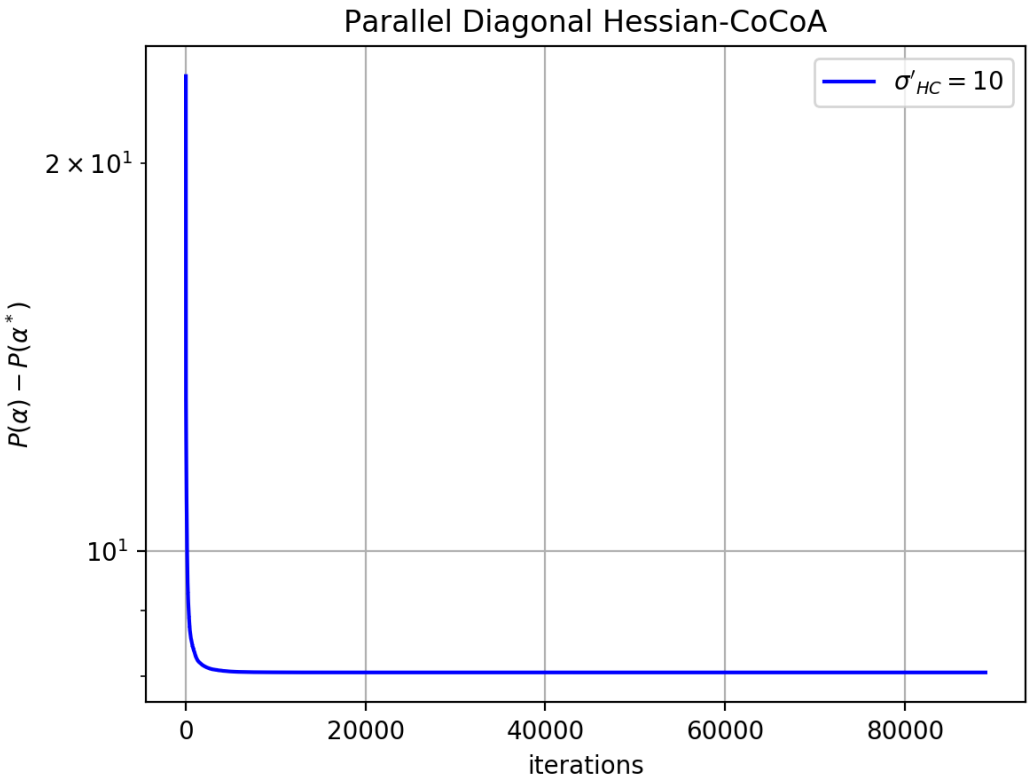


Figure 5.3: duke data set, duality gap and primal suboptimality respectively vs. number of iterations, $n = 7.129$, $d = 44$, $P = 100$.

Chapter 6

Conclusion and Future Directions

In this work we have deeply studied CoCoA, an existing distributed framework for empirical risk minimization problems with general non-strongly convex regularizers. Even though the original problems addressed by CoCoA are not separable, the computation is splitted among machines by the definition of local subproblems and the introduction of the parameter σ'_C. Starting from this analysis and motivated by the need of improving performances in cases such as logistic regression loss, we have modified the structure of local subproblems: the main result of adopting a better local model is to ensure faster convergence rate for the critical cases of CoCoA. This new distributed framework, Hessian-CoCoA, has been shown to be nothing but a generalization of its ancestor: indeed, CoCoA has been derived as a special instance. In conclusion, we have also extended our analysis to the parallel framework: both CoCoA and Hessian-CoCoA are indeed also suited for a parallel setting. Starting from the state-of-art parallel algorithms over columns, we have assumed to work with a particular instance of Hessian-CoCoA, where the local subproblems are one dimensional. In partcular, we have derived a runtime guarantee which proves that our algorithm applied in the parallel framework is competitive with the state-of-art. In the following sections, we briefly analyze some of the major directions that are worth to be considered in a future work.

6.1 Parallel Diagonal Hessian-CoCoA with Blocks

In order to reduce the lower bound of σ'_{HC}, and, therefore, improve the convergence rate also for datasets with strongly correleted features, we could try to introduce an additional parameter: following the main idea behind Block-Greedy Coordinate Descent framework, togehter with the degree of parallelization P, we could also consider the features to be clustered in B blocks, see [3]. In particular, in each iteration, the algorithm would select

uniformly at random at most one feature per block, for a total number of P features that will be processed in parallel. Consequently, the range for those parameters are the following:

$$B \subseteq \{1, \dots, n\}\,,$$
$$P \subseteq \{1, \dots, B\}\,.$$

6.1.1 Block Spectral Radius

We now define an important parameter, which is also the main element to be taken into account for preprocessing the features and clustering them into separate blocks:

$$\rho_{block} := \max_{M \in \mathcal{M}} \rho(\mathcal{M})\,,$$

where $\mathcal{M}$ is the set of all $B \times B$ matrices that are obtained from $A^\top A$ by selecting exactly one index from each of the B blocks.

6.1.2 Feature Clustering

The main reason for adopting a feature clustering preprocessing of the data is to minimize the block spectral radius, and, therefore, the safe lower bound of σ'_{HC}: indeed, by clustering highly correlated features within the same block, ideally, we would achieve that features belonging to different blocks are almost orthogonal, resulting in $\rho_{block} \approx 1$. Of course, directly finding a clustering that minimizes this quantity is a computationally daunting task, given the enormous space of possible partitions. But efficient heuristics have been explored, which focuses mainly on minimizing the inner product of features from different blocks. The following lemma explains the idea behind these heuristics.

Lemma 6.1 *(see [3]). Let $S \in \mathbb{R}^{B \times B}$ be positive semidefinite, with $S_{i,i} = 1$, and $|S_{i,j}| < \varepsilon$ for $i \neq j$. Then the spectral radius of S has the upper bound*

$$\rho(S) \leq 1 + (B-1)\varepsilon\,.$$

Proof Let x be the eigenvector corresponding to the largest eigenvalue of S, scaled so that $\|x\|_1 = 1$. Then

$$\rho(S) = \|Sx\|_1 = \sum_i \left| x_i + S_{i,j} \sum_{j \neq i} x_j \right| \leq \sum_i \left(|x_i| + \varepsilon \sum_{j \neq i} |x_j| \right) = 1 + (B-1)\varepsilon\,. \tag{6.1}$$

□

Indeed, as this proposition confirms, it is reasonable to adopt an heuristic approach that aim at minimizing the maximum absolute inner product bewteen features (column of the data matrix).

A good example is the heuristic for uniform-sized clusters proposed by Scherrer, Tewari, Halappanavar and Haglin in [3], which requires a computation time of $O(Bn)$.

6.2 Mini-Batches Parallelization

Another way to minimize the σ'_{HC} parameter, would be to exploit a better model in each round. Proceeding in this direction, we propose to extend the parallel analysis to a mini-batches structure. In particular, in a mini-batches parallel framework, given P processors working in parallel, each of them optimizes a mini-batch subset $\mathcal{P}_k$ of the variables, where

$$\sum_{k=1}^{P} |\mathcal{P}_k| = P_t << n\,,$$

and P_t is the total number of variables optimized in parallel in one round. The subproblem solved in iteration t by processor k is

$$\Delta\boldsymbol{\alpha}_{[k]} = \underset{\Delta\boldsymbol{\alpha}\in\mathbb{R}^{|\mathcal{P}_k|}}{\arg\min} \left\{ \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A_{[k]}\Delta\boldsymbol{\alpha} + \frac{\sigma'_{HC}}{2}\Delta\boldsymbol{\alpha}^\top A_{[k]}^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A_{[k]}\Delta\boldsymbol{\alpha} + \lambda \sum_{i\in\mathcal{P}_k} |\alpha_i^{(t)} + \Delta\alpha_i| \right\},$$

globally resulting in the following optimization problem

$$\Delta\boldsymbol{\alpha} = \underset{\Delta\boldsymbol{\alpha}\in\mathbb{R}^{P_t}}{\arg\min} \left\{ \sum_{k=1}^{P} \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A_{[k]}\Delta\boldsymbol{\alpha}_{[k]} + \frac{\sigma'_{HC}}{2} \sum_{k=1}^{P} \Delta\boldsymbol{\alpha}_{[k]}^\top A_{[k]}^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A_{[k]}\Delta\boldsymbol{\alpha}_{[k]} + \lambda\|\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}\| \right\}.$$

The advange of a mini-batch approach is that we achive convergence for smaller values of σ'_{HC} w.r.t. the parallel diagonal version: indeed, the local model is more refined. On the other hand, subproblems are computationally more expensive w.r.t. the diagonal Hessian-CoCoA framework. This configuration could be particularly convenient when $\rho(A^\top A) >> 1$: to better capture the correlation between features, it is important to work with more refined subproblems, that also considers the crossed second derivatives.

6.2.1 Block-Structure

In order to reduce σ'_{HC} even more, similarly to what has been proposed for the parallel diagonal Hessian-CoCoA version, we could add to the framework another level of abstraction. Following the main idea behind Block-Greedy Coordinate Descent framework and motivated from the need of good performances even for data set with $\rho(A^\top A) >> 1$, we could try to

introduce an additional parameter: togehter with the degree of parallelization P, we could also consider the features to be clustered in B blocks. In particular, in each iteration, the algorithm selects uniformly at random P different blocks. Then, each processor k is assigned with a distinct block, from which it selects uniformly at random $|\mathcal{P}_k|$ features. Assuming the blocks to be equally sized, the total number of blocks and their size should be in the following ranges:

$$B \in \{P, \dots, n\}$$
$$|B| \geq \max_{k \in P} |\mathcal{P}_k| .$$

The idea behind the introduction of the block structure, is to reduce as much as possible the correlation between features that are operated by different processors. Indeed, this will result in a smaller value of σ'_{HC}, and, ideally, we could obtain that features belonging to different blocks are independent.

Appendix A

Appendix

A.1 Convex Conjugates

Lemma A.1 *(Conjugate of Logistic Loss). The logistic classifier loss function*

$$f(A\boldsymbol{\alpha}) := \sum_{j=1}^{d} \log\left(1 + \exp(-b_j \mathbf{y}_j^\top \boldsymbol{\alpha})\right) , \tag{A.1}$$

is the conjugate of f^*, where

$$f^*(\mathbf{w}) := \sum_{\substack{j=1 \\ -w_j b_j \neq 1}}^{d} (1 + w_j b_j)\log(1 + w_j b_j) - \sum_{\substack{j=1 \\ -w_j b_j \neq 0}} w_j b_j \log\left(-w_j b_j\right) , \tag{A.2}$$

with the box constraint $-w_j b_j \in [0, 1]$.

Proof By separability of f^*, the conjugate of $f^*(\mathbf{v}) = \sum_{j=1}^{d} \Phi_j^*(v_j)$ is $f(\mathbf{w}) = \sum_{j=1}^{d} \Phi_j(w_j)$. Given the logistic loss, the conjugate pairs are $\Phi_j(u) = \log\left(1 + \exp(-b_j u)\right)$, and $\Phi_j^* = -w_j b_j \log\left(-w_j b_j\right) + (1 + w_j b_j)\log\left(1 + w_j b_j\right)$, with $-w_j b_j \in [0, 1]$.□

Lemma A.2 *(Conjugate of the Elastic Net Regularizer). For $\eta \in (0, 1]$, the elastic net function $g_i(\alpha) = \frac{\eta}{2}\alpha^2 + (1 - \eta)|\alpha|$ is the convex function of*

$$g_i^*(x) := \frac{1}{2\eta}\left([|x| - (1 - \eta)]_+\right)^2 . \tag{A.3}$$

In addition, g^* is smooth, i.e. it has Lipschitz continuous gradient with constant $\frac{1}{\eta}$.

Proof We start by applying the definition of convex conjugate, that is:

$$g^*(x) := \max_{\alpha \in \mathbb{R}} \left[x\alpha - \eta \frac{\alpha^2}{2} - (1-\eta)|\alpha| \right] . \tag{A.4}$$

We now proceed by distringuishing two cases:

1. $\alpha^\star \geq 0$
2. $\alpha^\star < 0$

For the first case, we get that

$$g^*(x) = \max_{\alpha \in \mathbb{R}} \left[x\alpha - \eta \frac{\alpha^2}{2} - (1-\eta)\alpha \right] . \tag{A.5}$$

Setting the derivative to zero, we get $\alpha^\star = \frac{x-(1-\eta)}{\eta}$. Then, in order to satisfy $\alpha^\star \geq 0$, we must have $x \geq 1-\eta$. Replacing $\alpha^\star$ we get

$$\begin{aligned} g^*(x) &= \alpha^\star(x - \frac{1}{2}\eta\alpha^\star - (1-\eta)) \\ &= \alpha^\star(x - \frac{1}{2}(x-(1-\eta)) - (1-\eta)) \\ &= \frac{1}{\eta}(x-(1-\eta))^2 . \end{aligned} \tag{A.6}$$

For the second case, we similarly get that for $x \leq -(1-\eta)$

$$g^*(x) = \frac{1}{2\eta}(x+(1-\eta))^2 . \tag{A.7}$$

Finally, by the fact that $g^*(\cdot)$ is convex, always positive, and $g^*(-(1-\eta)) = g^*(1-\eta) = 0$, it finally follows that $g^*(x) = 0 \; \forall x \in [-(1-\eta), 1-\eta]$. □

Lemma A.3 *(Conjugate of the modified L1-norm). The convex conjugate of the bounded support modification of the L1-norm, is*

$$\bar{g}_i^*(x) := \begin{cases} 0 & : \; x \in [-1,1] \\ B(|x|-1) & : \; \textit{otherwise} . \end{cases} \tag{A.8}$$

Proof We start by applying the definition of convex conjugate:

$$\bar{g}_i(\alpha) = \sup_{x \in \mathbb{R}^n} \left[\alpha x - \bar{g}_i^*(x) \right] . \tag{A.9}$$

We begin by looking at the case in which $\alpha \geq B$: it is easy to see that when $x \to +\infty$, we have:

$$\alpha x - B(|x| - 1) = (\alpha - B)x - B \to +\infty\,, \tag{A.10}$$

as $\alpha - B \geq 0$. The case $\alpha \leq -B$ holds analogously. Next, we analyze the case $\alpha \in [0, B]$: in this case it is clear that we have $\alpha^\star \geq 0$ and $\alpha^\star \leq 1$, since

$$\alpha x - B(x - 1) < \alpha x\,, \tag{A.11}$$

for every $x > 1$. Therefore the maximization becomes

$$\bar{g}_i(\alpha) = \sup_{x \in [0,1]} \alpha x\,, \tag{A.12}$$

which has its minimum α when $x = 1$. The dual case, when $\alpha \in [-B, 0]$, can be proven analogously. □

A.2 Convergence Result for Strongly Convex Regularizers

In the following section, the convergence results for strongly convex regularizers are exposed.

Lemma A.4 *Assume that $g_i(0) \in [0, 1]$ for all $i \in [n]$; then for the zero vector $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, we have*

$$\mathcal{O}_A(\boldsymbol{\alpha}^{(0)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star) = \mathcal{O}_A(\mathbf{0}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star) \leq n\,. \tag{A.13}$$

Proof By applying the definition of duality gap, we obtain

$$0 \leq \mathcal{O}_A(\mathbf{0}) - (-\mathcal{O}_B(\mathbf{w}(\mathbf{0}))) = f(\mathbf{0}) + f^*(\mathbf{w}(\mathbf{0})) + g(\mathbf{0}) + g^*(\mathbf{0})\,. \tag{A.14}$$

Since $\mathbf{w} := \nabla f(A\boldsymbol{\alpha})$, $f(\boldsymbol{\alpha}) + f^*(\mathbf{w}) = \boldsymbol{\alpha}^\top \mathbf{w}$, we can write

$$f^*(\mathbf{w}(\mathbf{0})) = f^*(\nabla f(\mathbf{0})) = \mathbf{0}^\top \nabla f(\mathbf{0}) - f(\mathbf{0}) = -f(\mathbf{0})\,. \tag{A.15}$$

Therefore, for $\boldsymbol{\alpha} = \mathbf{0}$, the duality gap reduces to

$$g(\mathbf{0}) + g^*(\mathbf{0}) \geq 0\,. \tag{A.16}$$

Therefore, given our initial assumption on g

$$-g^*(\mathbf{w}(\mathbf{0})) \leq \min g(\mathbf{0}) = 0\,, \tag{A.17}$$

which leads to the final inequality

$$0 \leq \mathcal{O}_A(\mathbf{0}) - (-\mathcal{O}_B(\mathbf{w}(\mathbf{0}))) \leq n\,. \tag{A.18}$$

□

Theorem A.5 *Assume that g_i are μ-strongly convex $\forall i \in [n]$. We define $\sigma_{max} = \max_{k\in[K]} \sigma_k$. Then, after T iterations of Algorithm 1, with*

$$T \geq \frac{1}{\gamma(1-\Theta)} \frac{\mu\tau + \sigma_{max}\sigma'}{\mu\tau} \log \frac{n}{\varepsilon_{\mathcal{O}_A}}\,, \tag{A.19}$$

it holds that

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(T)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star)\right] \leq \varepsilon_{\mathcal{O}_A}\,. \tag{A.20}$$

Furthermore, after T iterations with

$$T \geq \frac{1}{\gamma(1-\Theta)} \frac{\mu\tau + \sigma_{max}\sigma'}{\mu\tau} \log\left(\frac{1}{\gamma(1-\Theta)} \frac{\mu\tau + \sigma_{max}\sigma'}{\mu\tau} \frac{n}{\varepsilon_G}\right), \tag{A.21}$$

we have that the duality gap

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(T)}) - (-\mathcal{O}_B(\mathbf{w}(\boldsymbol{\alpha}^{(T)})))\right] \leq \varepsilon_G\,. \tag{A.22}$$

Proof Given that $g_i(\cdot)$ is μ-strongly convex w.r.t. the norm $\|\cdot\|$, and by applying the definition of σ_k, we obtain

$$\begin{aligned} R^{(t)} &\leq -\frac{\tau\mu(1-s)}{\sigma' s}\|\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}\|^2 + \sum_{k=1}^{K} \sigma_k \|\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}\|^2 \\ &\leq \left(-\frac{\tau\mu(1-s)}{\sigma' s} + \sigma_{max}\right) \|\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}\|^2\,, \end{aligned} \tag{A.23}$$

where $\sigma_{max} = \max_{k\in[K]} \sigma_k$. If we plug the following value of s

$$s = \frac{\tau\mu}{\tau\mu + \sigma_{max}\sigma'} \in [0,1] \tag{A.24}$$

into (A.23), we obtain that $R^{(t)} \leq 0 \ \forall t$. Still by considering the same value of s, we obtain

$$\begin{aligned}\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] &\geq \gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}G(\boldsymbol{\alpha}^{(t)}) \\ &\geq \gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}\left(\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})\right).\end{aligned} \tag{A.25}$$

Using the fact that $\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)} - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)}))\right] = \mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{\star}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] + \mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})$, we have

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{\star}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] + \mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star}) \geq \gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}\left(\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})\right), \tag{A.26}$$

which is equivalent to

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})\right] \leq \left(1 - \gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}\right)\left(\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})\right). \tag{A.27}$$

Therefore, denoting $\varepsilon_{\mathcal{O}_A}^{(t)} = \mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})$, we recursively have

$$\begin{aligned}\mathbb{E}\left[\varepsilon_{\mathcal{O}_A}^{(t)}\right] &\leq \left(1 - \gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}\right)^t \varepsilon_{\mathcal{O}_A}^{(0)} \\ &\leq \left(1 - \gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}\right)^t n \\ &\leq \exp\left(-t\gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}\right) n\,.\end{aligned} \tag{A.28}$$

The right hand side is smaller than some $\varepsilon_{\mathcal{O}_A}$ if

$$t \geq \frac{1}{\gamma(1-\Theta)}\frac{\tau\mu+\sigma_{max}\sigma'}{\tau\mu}\log\left(\frac{n}{\varepsilon_{\mathcal{O}_A}}\right). \tag{A.29}$$

To bound the duality gap in addition we have that

$$\gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}G(\boldsymbol{\alpha}^{(t)}) \leq \mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t+1)})\right] \leq \mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})\right]. \tag{A.30}$$

Thus, $G(\boldsymbol{\alpha}^{(t)}) \leq \frac{1}{\gamma(1-\Theta)} \frac{\tau\mu+\sigma_{max}\sigma'}{\tau\mu} \varepsilon_{\mathcal{O}_A}^{(t)}$. Hence, if $\varepsilon_{\mathcal{O}_A} \leq \gamma(1-\Theta)\frac{\tau\mu}{\tau\mu+\sigma_{max}\sigma'}\varepsilon_G$, then $G(\boldsymbol{\alpha}^{(t)}) \leq \varepsilon_G$. Therefore, after

$$t \geq \frac{1}{\gamma(1-\Theta)} \frac{\tau\mu+\sigma_{max}\sigma'}{\tau\mu} \log\left(\frac{1}{\gamma(1-\Theta)} \frac{\tau\mu+\sigma_{max}\sigma'}{\tau\mu} \frac{n}{\varepsilon_G}\right) \tag{A.31}$$

iterations, we obtain that the duality gap is less that ε_G. □

A.3 Runtime Guarantee for Stochastic CDN

The following theorem establishes runtime guarantee for stochastic CDN.

Theorem A.6 *Let $\boldsymbol{\alpha}^\star$ be a minimizer of $\mathcal{O}_A(\boldsymbol{\alpha}) = f(A\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha}\|_1$, where the function f is $\frac{1}{\tau}$-smooth and twice differentiable. Let $\boldsymbol{\alpha}^{(T)}$ denote the weight vector $\boldsymbol{\alpha}$ at the end of iteration T of stochastic CDN. Then,*

$$\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(T)})\right] - \mathcal{O}_A(\boldsymbol{\alpha}^\star) \leq \frac{n\Psi(0)}{T+1} \tag{A.32}$$

where n is the number of features and

$$\Psi(\boldsymbol{\alpha}) = \frac{1}{2\tau}\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}\|_2^2 + \mathcal{O}_A(\boldsymbol{\alpha})\,. \tag{A.33}$$

Proof We start by bounding above the stochastic CDN update

$$\begin{aligned}\Delta\alpha_i &= \arg\min_{\Delta\alpha\in\mathbb{R}} \left\{ (\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i \Delta\alpha + \frac{1}{2}\Delta\alpha^\top (A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A)_{[i,i]} \Delta\alpha + \lambda|\alpha_i^{(t)} + \Delta\alpha| \right\} \\ &\leq \arg\min_{\Delta\alpha\in\mathbb{R}} \left\{ (\nabla f(A\boldsymbol{\alpha}^{(t)})^\top A)_i \Delta\alpha + \frac{1}{2\tau}\Delta\alpha^\top (A^\top A)_{[i,i]} \Delta\alpha + \lambda|\alpha_i^{(t)} + \Delta\alpha| \right\}\end{aligned} \tag{A.34}$$

Then, we analyze $\Delta\alpha_i$, where $\Delta\alpha_i = S_{\frac{\lambda}{\frac{1}{\tau}}}(\alpha_i - \frac{(\nabla f(A\boldsymbol{\alpha})^\top A)_i}{\frac{1}{\tau}}) - \alpha_i$. This is equivalent to claim that

$$\Delta\alpha_i = \arg\min_{\Delta\alpha\in\mathbb{R}} \left\{ (\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i \Delta\alpha + \frac{1}{2\tau}(\Delta\alpha)^2 + \lambda|\alpha_i^{(t-1)} + \Delta\alpha| \right\}. \tag{A.35}$$

Indeed, the optimality condition for the above optimization problem is

$$0 = (\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i + \frac{1}{\tau}\Delta\alpha + \lambda\nu_i\,, \tag{A.36}$$

where $\nu_i \in \delta|\alpha_i^{(t-1)} + \Delta\alpha|$ is the subdifferential of the absolute value at $\alpha_i^{(t-1)} + \Delta\alpha$. Given that $\nu_i = sign(\alpha_i^{(t-1)} + \Delta\alpha)$ if $\alpha_i^{(t-1)} + \Delta\alpha \neq 0$, and $\nu_i \in [-1, 1]$ otherwise, we have the following three cases:

- $\Delta\alpha > -\alpha_i^{(t-1)} \implies \nu_i = 1 \implies \Delta\alpha = \frac{-(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i - \lambda}{\frac{1}{\tau}} > -\alpha_i^{(t-1)}$,
- $\Delta\alpha < -\alpha_i^{(t-1)} \implies \nu_i = -1 \implies \Delta\alpha = \frac{-(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i + \lambda}{\frac{1}{\tau}} < -\alpha_i^{(t-1)}$,
- $\Delta\alpha = -\alpha_i^{(t-1)}$.

But this is nothing but the definition of $\Delta\alpha_i$.

We then proceed by defining the potential function

$$\Phi(\boldsymbol{\alpha}^{(t)}) = \frac{1}{2}\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\|_2^2, \tag{A.37}$$

and by bounding $\Delta_{t,\mathcal{P}_t} = \Phi(\boldsymbol{\alpha}^{(t-1)}) - \Phi(\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha})$, where $\Delta\alpha_i = 0 \; \forall i \notin \mathcal{P}_t$

$$\begin{aligned}
\Delta_{t,i} &:= \Phi(\boldsymbol{\alpha}^{(t-1)}) - \Phi(\boldsymbol{\alpha}^{(t)}) = \frac{1}{2}\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t-1)}\|_2^2 - \frac{1}{2}\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t-1)} - \Delta\boldsymbol{\alpha}\|_2^2 \\
&= \frac{1}{2}(\alpha_i^\star - \alpha_i^{(t-1)})^2 - \frac{1}{2}(\alpha_i^\star - \alpha_i^{(t-1)} - \Delta\alpha_i)^2 \\
&= \frac{1}{2}(\Delta\alpha_i)^2 + \frac{(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i}{\frac{1}{\tau}}(\alpha_i^{(t-1)} + \Delta\alpha_i - \alpha_i^\star) + \frac{\lambda\nu_i}{\frac{1}{\tau}}(\alpha_i^{(t-1)} + \Delta\alpha_i - \alpha_i^\star).
\end{aligned} \tag{A.38}$$

Since ν_i is a subdifferential, it follows that

$$\nu_i(\alpha_i^{(t-1)} + \Delta\alpha_i - \alpha_i^\star) \leq |\alpha_i^{(t-1)} + \Delta\alpha_i| - |\alpha_i^\star|. \tag{A.39}$$

We also have that

$$f(A\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha}) - f(A\boldsymbol{\alpha}^{(t-1)}) \leq (\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)\Delta\boldsymbol{\alpha} + \frac{1}{2\tau}(\Delta\boldsymbol{\alpha})^\top(\Delta\boldsymbol{\alpha}). \tag{A.40}$$

Consequently,

$$\begin{aligned}
\Delta_{t,i} \geq \; & \tau\left(f(A\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha}) - f(A\boldsymbol{\alpha}^{(t-1)})\right) + \\
& + \frac{(\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A)_i}{\frac{1}{\tau}}(\alpha_i^{(t-1)} - \alpha_i^\star) + \frac{\lambda}{\frac{1}{\tau}}(|\alpha_i^{(t-1)} + \Delta\alpha_i| - |\alpha_i^\star|).
\end{aligned} \tag{A.41}$$

Taking the expectation value, conditioned of $\boldsymbol{\alpha}^{(t-1)}$, we obtain that

$$\begin{aligned}
&\mathbb{E}\left[\Phi(\boldsymbol{\alpha}^{(t-1)}) - \Phi(\boldsymbol{\alpha}^{(t)})\right] = \mathbb{E}_i\left[\Delta_{t,i}\right] = \frac{1}{n}\sum_{i=1}^{n}\Delta_{t,i} \\
&\geq \frac{\tau}{n}\left[\sum_{i=1}^{n}\left(f(A\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha}\mathbf{e}_i) - f(A\boldsymbol{\alpha}^{(t-1)})\right)\right] + \frac{\tau}{n}\nabla f(A\boldsymbol{\alpha}^{(t-1)})^\top A(\boldsymbol{\alpha}^{(t-1)} - \boldsymbol{\alpha}^\star) + \\
&+ \frac{\tau}{n}\lambda\sum_{i=1}^{n}\left(|\alpha_i^{(t-1)} + \Delta\alpha_i| - |\alpha_i^\star|\right) \\
&\geq \frac{\tau}{n}\left[\sum_{i=1}^{n}\left(f(A\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha}\mathbf{e}_i) - f(A\boldsymbol{\alpha}^{(t-1)})\right)\right] + \frac{\tau}{n}\left[f(A\boldsymbol{\alpha}^{(t-1)}) - f(A\boldsymbol{\alpha}^\star)\right] + \\
&+ \frac{\tau}{n}\lambda\sum_{i=1}^{n}\left(|\alpha_i^{(t-1)} + \Delta\alpha_i| - |\alpha_i^\star|\right) \\
&= \tau\left[\mathbb{E}\left[f(A\boldsymbol{\alpha}^{(t)}) - f(A\boldsymbol{\alpha}^{(t-1)})\right] + \frac{(f(A\boldsymbol{\alpha}^{(t-1)}) - f(A\boldsymbol{\alpha}^\star))}{n} + \frac{\lambda}{n}\sum_{i=1}^{n}|\alpha_i^{(t-1)} + \Delta\alpha_i| - \lambda\frac{\|\boldsymbol{\alpha}^\star\|}{n}\right]
\end{aligned} \tag{A.42}$$

Note that, we have

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{\alpha}^{(t)}\|_1\right] &= \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{\alpha}^{(t-1)} + \Delta\boldsymbol{\alpha}\mathbf{e}_i\|_1 \\
&= \|\boldsymbol{\alpha}^{(t-1)}\|_1 - \frac{1}{n}\|\boldsymbol{\alpha}^{(t-1)}\|_1 + \frac{1}{n}\sum_{i=1}^{n}|\boldsymbol{\alpha}_i^{(t-1)} + \Delta\alpha_i|\,.
\end{aligned} \tag{A.43}$$

Plugging this in the above equation, gives us

$$\begin{aligned}
&\frac{1}{\tau}\mathbb{E}\left[\Phi(\boldsymbol{\alpha}^{(t-1)}) - \Phi(\boldsymbol{\alpha}^{(t)})\right] \\
&\geq \mathbb{E}\left[f(A\boldsymbol{\alpha}^{(t)}) + \lambda\|\boldsymbol{\alpha}^{(t)}\|_1 - f(A\boldsymbol{\alpha}^{(t-1)}) - \lambda\|\boldsymbol{\alpha}^{(t-1)}\|_1\right] + \\
&+ \frac{(f(A\boldsymbol{\alpha}^{(t-1)}) + \lambda\|\boldsymbol{\alpha}^{(t-1)}\|_1 - f(A\boldsymbol{\alpha}^\star) - \lambda\|\boldsymbol{\alpha}^\star\|_1)}{n} \\
&= \mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)})\right] + \frac{(\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star))}{n}\,.
\end{aligned} \tag{A.44}$$

This is equivalent to

$$\mathbb{E}\left[\frac{1}{\tau}\Phi(\boldsymbol{\alpha}^{(t-1)}) + \mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \frac{1}{\tau}\Phi(\boldsymbol{\alpha}^{(t)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{(t)})\right] \geq \frac{(\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^\star))}{n}\,. \tag{A.45}$$

We now define the composite potential function

$$\Psi(\boldsymbol{\alpha}) = \frac{1}{\tau}\Phi(\boldsymbol{\alpha}) + \mathcal{O}_A(\boldsymbol{\alpha})\,, \tag{A.46}$$

and, taking full expectation, we get

$$\mathbb{E}\left[\Psi(\boldsymbol{\alpha}^{(t-1)}) - \Psi(\boldsymbol{\alpha}^{(t)})\right] \geq \frac{1}{n}\mathbb{E}\left[\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})\right]\,. \tag{A.47}$$

Summing over $t = 1, \ldots, T$ and realizing that $\mathcal{O}_A(\boldsymbol{\alpha}^{(t)})$ monotonically decreases, we obtain

$$\begin{aligned}\frac{1}{n}\mathbb{E}\left[(T+1)(\mathcal{O}_A(\boldsymbol{\alpha}^{(T)})) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star})\right] &\leq \frac{1}{n}\mathbb{E}\left[\sum_{t=1}^{T+1}(\mathcal{O}_A(\boldsymbol{\alpha}^{(t-1)}) - \mathcal{O}_A(\boldsymbol{\alpha}^{\star}))\right] \\ &\leq \mathbb{E}\left[\sum_{t=1}^{T+1}(\Psi(\boldsymbol{\alpha}^{(t-1)}) - \Psi(\boldsymbol{\alpha}^{(t)}))\right] \\ &= \mathbb{E}\left[(\Psi(\mathbf{0}) - \Psi(\boldsymbol{\alpha}^{T+1}))\right] \leq \mathbb{E}\left[\Psi(\mathbf{0})\right] = \Psi(\mathbf{0})\,.\end{aligned} \tag{A.48}$$

□

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Bibliography

[1] Francis Bach. Self-concordant analysis for logistic regression. *Electronic Journal of statistics*, 4:384–414, 2010.

[2] Martin Takáč Martin Jaggi Celestine Dünner, Simone Forte. Primal-dual rates and certificates. *ICML - Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA*, 48:783–792, 2016.

[3] Mahantesh Halappanavar David Haglin Chad Scherrer, Ambuj Tewari. Feature clustering for accelerating parallel coordinate descent. *Neural Information Processing Systems Foundations*, 2012.

[4] Cho-Jui Hsieh and Inderijit S. Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

[5] D. Bickson J. K. Bradley, A. Kyrola and C. Guestrin. Parallel coordinate descent for $\ell 1$-regularized loss minimization. *International Conference on Machine Learning*, 2011.

[6] Stephen Boyd Kwangmoo Koh, Seung-Jean Kim. An interior-point method for large scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

[7] Martin Takac jonathan Terhorst Sanjay Krishnan Thomas Hofmann Micheal I. Jordan Martin Jaggi, Virgina Smith. Communication-efficient distributed dual coordinate ascent. *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014.

[8] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.

[9] J. Nocedal and S. Wright. *Numerical optimization*. Springer Series in Operations Research, 1999.

[10] S. Yun P. Tseng. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, 140:513–535, 2009.

[11] Ambuj Tewari Shai Shalev-Shwartz. Stochastic methods for ℓ_1-regularized loss minimization. *The Journal of Machine Learning*, 12:1865–1892, 2011.

[12] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14:567–599, 2013.

[13] Chenxin Ma Martin Takac Michael I. Jordan Martin Jaggi Virginia Smith, Simone Forte. Cocoa: a general framework for communication-efficient distributed optimization. 2016. Available at https://arxiv.org/abs/1611.02189.

[14] Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151:3–34, 2015.

[15] Yuncai Liu Ming-Hsuan Yang Yatao Bian, Xiong Li. Parallel coordinate descent newton method for efficient ℓ_1-regularized minimization. 2013. Available at https://arxiv.org/abs/1306.4080.