Constrained machine learning: algorithms and models

Geoffrey Negiar

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-229
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-229.html

August 20, 2023

Constrained Machine Learning: Algorithms and Models

by

Geoffrey Négiar

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Laurent El Ghaoui, Co-chair
Professor Michael Mahoney, Co-chair
Professor Somayeh Sojoudi
Assistant Professor Aditi Krishnapriyan

Summer 2023

Constrained Machine Learning: Algorithms and Models

Abstract

Constrained Machine Learning: Algorithms and Models

by

Geoffrey Négiar

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Science

University of California, Berkeley

Professor Laurent El Ghaoui, Co-chair

Professor Michael Mahoney, Co-chair

This thesis is concerned with designing efficient methods to incorporate known structure in machine learning models. Structure arises either from problem formulation (e.g. physical constraints, aggregation constraints), or desirable model properties (energy efficiency, sparsity, robustness). In many cases, the modeler has a certain knowledge about the system that they are modeling, which must be enforced in an exact manner. This can be necessary for providing adequate safety guarantees, or for improving the system's efficiency: training a system with less data, or less computation costs. This thesis provides methods to do so in a variety of settings, by building on the two foundational fields of continuous, constrained optimization and of differentiable statistical modeling (also known as deep learning).

The first part of the thesis is centered on designing and analyzing efficient algorithms for optimization problems with convex constraints. In particular, it focuses on two variants of the Frank-Wolfe algorithm: the first variant proposes a fast backtracking-line search algorithm to adaptively set the step size in the full-gradient setting; the second variant proposes a fast stochastic Frank-Wolfe algorithm for constrained finite-sum problems. I also describe contributions to open-source constrained optimization software. The second part of this thesis is concerned with designing deep learning models which enforce certain constraints exactly: constraints based on physics, and aggregation constraints for probabilistic forecasting models. This part leverages bi-level optimization models, and differentiable optimization to constrain the output of a complex neural network. We demonstrate that complex non-linear constraints can be enforced on complex non-convex models, including probabilistic models.

These examples showcase the power of hybrid models which couple data-driven learning and leverage complex nonlinear models such as deep neural networks, and well-studied optimization problems allowing for efficient algorithms. These hybrid models help highly flexible models to pick up structural patterns, achieving strong performance with sometimes no data access at all.

To my grandparents, Jeannine, Charles, Louise and James.
To my parents, Carol and Xavier.

Contents

List of Figures

List of Tables

Acknowledgments

My PhD adventure would have been nothing without all the people who have supported me over the years, during the PhD itself of course, but also way before, paving my way here. I dedicate this thesis to my grandparents, and my parents. Jeannine, who I never got to meet, but who indirectly defined so much of who I am. Charles, his survivor mindset and endless curiosity. Louise, for being my anchor in California, with Uncle Gerry and Aunt Gail. James, for getting me started in engineering, and the American practical spirit through many (many) hardware store hangouts. Mom, Dad, thank you for your kindness, your open minds, and for aiming to pass on your ability to be everywhere at home. Thank you also to my cousins, Laure, Julie-Anne, Edouard and Claire, and to my Aunt Karen and Uncle Didier for making growing up as an American in Paris seem completely normal – also for preceding me to the grand California adventure, letting me stay, and help me be at home in San Francisco and New York. Ed, thanks for sharing all your favorite places in SF – I would not know half the restaurants and bars I know without you. Also, thank you for introducing me to great people like Jacob K, Germain B, Irva, thanks for the apple pie and more recs.

This thesis would never have happened if my advisors Laurent El Ghaoui and Michael Mahoney hadn't taken a chance on me. Thank you Laurent for teaching me the depths of optimization, and for many blackboard sessions figuring out how to model problems, and solve them fast. When I decided to switch gears a bit and focus on bringing structure to deep learning, Michael gave me the time and opportunity to learn, meet many new people and get a deep overview of ML based in the physical world. I'm very grateful for that.

Thank you to Fabian Pedregosa for being my mentor in all things open-source, continuous optimization, for mentoring me to write my first first-author paper, and for hosting me at Google. I learned so much! I hope that we'll go soon for the famed chocolate factory tour on the banks of the Léman. Aditi, that paper was a lot of fun, and I wish you all the best for the hard path of professorship. Thank you Rob Freund, for hopping in our paper, working on a proof together, and for your kindness and mentorship since. Thanks to Thomas Kerdreux, Gautier Gidel for keeping the Frank-Wolfe fire ablaze.

Shirley Salanio, you deserve special thanks for cheerfully supporting me (and all the other EECS students!) up until graduation. No-one would ever graduate without you.

I was lucky to be hosted in several internships before and during my PhD. Thank you to all the teams at Bloomberg LP, Google and Amazon for hosting me, helping me on all my Python, PyTorch, Jax and ML questions. In particular, Ryan T. Hoens, Kang Sun, Danny Tarlow, Daniel D. Johnson, Hugo Larochelle, Jake Vanderplas, Roy Frostig, Stephan Hoyer, Matthieu Blondel, Mengfei Cao, Ruijun Ma, Youxin Shen, Danielle Maddix, Kenny Shirley. Also my co-interns in NYC Joachim, Ege, Arturo, Matthieu (DRM), Hanjing, Bella, Léa, Patricia, Victoria, Simon R: you all made those summers fun and interesting (sometimes more productive, sometimes less ;)). Thanks also to all my Berkeley PhD friends, Armin, Nilesh, Allan, Morris, Daniel Rothschild, You Sun, Nilandri, Aldo, Colorado, Chenling, Esther, Neil, Eric Mazumdar. I loved sharing our challenges, discoveries, internship and job tips together. Finally, all the people I got to meet and hang out with during ML conferences: those were amongst the best times of my PhD!

I'd like to give thanks to the open-source teams who supported me during this journey: PyTorch, Jax, NumPy, SciPy, Sklearn. You make all of this possible, and practical. May you always be supported to do so.

I would not have gone the PhD route without my mentors at Shift Technology Alice (more to be said here :)), and Éric S. DJ, that burrito changed my academic trajectory. Thank you.

I had the great joy and opportunity to live in a shared house at California St in Berkeley with wonderful housemates, both for a long time and a short time. Romain, we managed to nurture a special environment over the years; thank you for reaching out to me ("Dear Kung Fu Master")! I'm glad Anna and Alice joined the ride for a bit. Philippe, Chiraz, Paul-Armand, Georg, Maddie, Hector, Julia and Titouan (j'ai hésité sur l'ordre du trio ;)), Heather, Khalil, Louise, Max F, but also honorary housemates Sevan, Thomas, Valentine, Ronan, Anne-Claire, Étienne, Hugo, Alexis, Grégoire Hamel, and Grégoire Mialon (for inaugurating the house together), Denis... quelle aventure! Thankgsgiving dinners or trips to LA, DIY house improvement (I still don't see a farmbot, but it'll happen someday), cocktail parties, treasure hunts, crazy snowshoeing for my poor urban self, music Wednesdays... More recently, thanks to the SF brewery night crew for being there in the last months of my thesis: David, Zack, Francesco, Rigney, Felix, Johanna, Eugene, Prastuti, Alex, Maud, Thibault. Talking about LA, thank you to my Angelenos brothers, Max and Adam G, and their parents, Dr. Russ and Wendy. Thanks to Louise F, Emma G and Noémie M for welcoming me in your unforgettable Rwanda road trip.

A big thanks to the French community in the SF Bay Area for being so French and so unFrench at the same time. Y'all feel like home in a good, weird, neither here nor there, way. Christelle, Antoine D (and Will!), Nicolas Zweibaum (from reading my CV in my application to many evenings of commiserating our LDRs), Matthieu, Marie, Johan, Nina, Raphael (who followed me into boxing for a bit), Ferdi, Jonas, Balthazar, Guillaume, Olivier, Solenne, Clément, Solène, Adrien, Pierre Fredenucci, Taha, Loic, Jérome T, Aurian & Antoine K, Ahmed EA, Ilai D, Charlotte W and Tom B, and many others.

Thanks to all my friends in France (and everywhere) who tolerated me only popping by occasionally, rarely announcing myself more than a few hours in advance ("Yo! Je suis à Paris, et toi?" or "Hey!! I'm in LA/NYC next week, hbu?"). Those who came visit in California and those who didn't (Covid19, I'm looking at you). Juliette, Alex Benoliel, Antoine DVD, Vincent, Djerby, Goga, Hugo, Sabrina, Jean-Baptiste L, Dhruv, Clara, Gregoire M, Yana, Antoine Gondé and Cassandre, Marc Szafraniec, Margot L, Solène, toute l'oenologie (Ambroise, Ludo, Pauline – merci pour ta patience :), Nico & Mathilde, Gabriel, Simon, Pierre), Jose, la HX2 (Christian, Nathan, Lucas, Jack, Majdi, Rémi, Alex Darmon, Antoine Lagarde, Victoire, Antoine Pelletier...). Thank you to the always-on, always active tech (and more) support line from les Bolosses.

Thank you to my martial arts families over the years, from boxing (Coach Jon and Coach Evan) to Systema (Juan, Patrick), my instructors at l'École polytechnique, the majors Sacha Engel and Alexandre Arisso, and Jujitsu Pariset. Martial arts have always kept me sane, and grounded me in times of need. From Cal Boxing: Jimmy, Kipper, Jacquie, Vivian, Eric, Gabriel R, and all the others: I boxed a bit too much in my early PhD years – it was a great

outlet from research. I'm glad I did though, and glad I stopped. Go Bears!

My New York adventure friends: Vincent B, Pierre-Louis C, Marc T, Yoann LC, Céline G, Nico, Camille S, Alexia P, Morgane, Julien GC, Léonore C.

Last but not least: Alice, you unwittingly got me started on this path, and you brought me to its conclusion. Thank you for your love; I can't wait to live all the years and moments to come.

I'm so grateful to share my time on Earth with you all. Thanks to all who have taken the time to share their essential selves with me, and to grow together, in big or small ways. Now, onward: to all the future moments together and with new people, with intensity and intent! If I've forgotten you, send me a text, and let's go for an adventure!

Chapter 1

Overview

In recent years, machine learning models have achieved countless success stories in fields aiming to match human perception (computer vision, audio processing, natural language). This success was achieved by understanding how to exploit structure in model inputs: digital representations of images, sound, text, code, and even molecules [1, 2, 3, 4]. To achieve similar levels of success in engineering and the sciences, models must incorporate additional structural constraints: both the model internals and the model outputs should satisfy certain key properties (e.g. sparse or low rank weights for model internals and physical equations for model outputs). Although the field of optimization has long been concerned with developing methods to enforce such constraints, efforts to tie in structure brought by optimization methods and the flexibility of data-driven models are very recent [5, 6]. This thesis proposes novel, efficient methods for incorporating structure in machine learning models, both in the model internals (Part I), and in the model outputs (Part II). We argue that such hybrid systems will be key to develop high performance systems for complex physical applications.

Structured constraints in Machine Learning have recently brought the Frank-Wolfe (FW) family of algorithms back in the spotlight. The Frank-Wolfe algorithm allows to enforce convex constraints on a decision variable (e.g. model weights), while maintaining a sparse representation of the decision variable. The first part of this thesis develops novel variants of the Frank-Wolfe algorithm to improve practical speed of the algorithm. Additionally, we describe our two open-source optimization libraries: COPT and CHOP.

When deploying decision-making systems in the wild, the systems must enforce physical constraints: discrepancies can lead to undefined decisions. For example, if we predict inbound water flow in reservoirs in a region at different granularities, the predictions at the different levels must enforce conservation of mass; otherwise, there will be quantities of water that are unaccounted for, breaking the decision making system. The second part of this thesis considers the problem of incorporating physical constraints into deep learning models, in the form of partial differential equations and hierarchical conservation of mass.

We now describe each section in more detail.

Backtracking line search for Frank-Wolfe (Chapter 2)

While the classical FW algorithm has poor local convergence properties, the Away-steps and Pairwise FW variants have emerged as improved variants with faster convergence. However, these improved variants suffer from two practical limitations: they require at each iteration to solve a 1-dimensional minimization problem to set the step-size and also require the Frank-Wolfe linear subproblems to be solved exactly. In this paper, we propose variants of Away-steps and Pairwise FW that lift both restrictions simultaneously. The proposed methods set the step-size based on a sufficient decrease condition, and do not require prior knowledge of the objective. Furthermore, they inherit all the favorable convergence properties of the exact line-search version, including linear convergence for strongly convex functions over polytopes. Benchmarks on different machine learning problems illustrate large performance gains of the proposed variants. Our backtracking line search scheme has become standard in the subsequent FW literature, as evidenced in the recent Frank-Wolfe book [7]. Code is available in our open source COPT library.

This work appears as Pedregosa et al. [8].

Stochastic Frank-Wolfe for constrained finite-sum minimization (Chapter 3)

Machine learning problems are often expressed as empirical risk minimization problems over a fixed training dataset. In this case, the objective is written as a finite-sum of loss terms over the datapoints in the dataset. Since it may be computationally intractable to keep the full dataset in memory, or to compute gradient of the full objective, it is common in the ML optimization literature to provide stochastic variants of optimization algorithms, where only sampled datapoints are used at each iteration. We provide an efficient stochastic Frank-Wolfe algorithm for these finite-sum problems, which can exploit sampling batches of arbitrary, fixed size. We provide rates of convergence for our algorithm in the convex objective case, and prove convergence in the nonconvex case. Code is available in our open source COPT library.

This work appears as Negiar et al. [9].

COPT and CHOP: open-source optimization libraries (Chapter 4)

In this chapter, we describe our software libraries for first order optimization methods: the COPT and CHOP libraries. We provide efficient implementations of state of the art constrained or composite optimization algorithms in NumPy (COPT) and PyTorch (CHOP).

Learning differentiable solvers for systems with hard constraints (Chapter 5)

We introduce a practical method to enforce partial differential equation (PDE) constraints for functions defined by neural networks (NNs), with a high degree of accuracy and up to a desired tolerance. We develop a differentiable PDE-constrained layer that can be incorporated into any NN architecture. Our method leverages differentiable optimization and the implicit function theorem to effectively enforce physical constraints. Inspired by dictionary learning, our model learns a family of functions, each of which defines a mapping from PDE parameters to PDE solutions. At inference time, the model finds an optimal linear combination of the functions in the learned family by solving a PDE-constrained optimization problem. Our method provides continuous solutions over the domain of interest that accurately satisfy desired physical constraints. Our results show that incorporating hard constraints directly into the NN architecture achieves much lower test error when compared to training on an unconstrained objective.

This work appears as Négiar, Mahoney, and Krishnapriyan [10].

Probabilistic forecasting with coherent aggregation (Chapter 6)

Obtaining accurate probabilistic forecasts while respecting hierarchical information is an important operational challenge in many applications, perhaps most obviously in energy management, supply chain planning, and resource allocation. The basic challenge, especially for multivariate forecasting, is that forecasts are often required to be coherent with respect to the hierarchical structure. In this chapter, we propose a new model which leverages a factor model structure to produce coherent forecasts by construction. This is a consequence of a simple (exchangeability) observation: permuting base-level series in the hierarchy does not change their aggregates. Our model uses a convolutional neural network to produce parameters for the factors, their loadings and base-level distributions; it produces samples which can be differentiated with respect to the model's parameters; and it can therefore optimize for any sample-based loss function, including the Continuous Ranked Probability Score and quantile losses. We can choose arbitrary continuous distributions for the factor and the base-level distributions. We compare our method to two previous methods which can be optimized end-to-end, while enforcing coherent aggregation. Our model achieves significant improvements: between $11.8 - 41.4\%$ on three hierarchical forecasting datasets. We also analyze the influence of parameters in our model with respect to base-level distribution and number of factors.

This work appears as Negiar et al. [11].

Part I

Algorithms

Chapter 2

Linearly convergent Frank-Wolfe with backtracking line-search

2.1 Introduction

The Frank-Wolfe (FW) or conditional gradient algorithm [12, 13, 14] is a method for constrained optimization that solves problems of the form

$$\underset{\boldsymbol{x}\in\operatorname{conv}(\mathcal{A})}{\text{minimize}}\ f(\boldsymbol{x})\,, \tag{OPT}$$

where f is a smooth function for which we have access to its gradient and $\operatorname{conv}(\mathcal{A})$ is the convex hull of $\mathcal{A}$; a bounded but potentially infinite set of elements in $\mathbb{R}^p$ which we will refer to as atoms.

The FW algorithm is one of the oldest methods for non-linear constrained optimization and has experienced a renewed interest in recent years due to its applications in machine learning and signal processing [15]. Despite some favorable properties, the local convergence of the FW algorithm is known to be slow, achieving only a sublinear rate of convergence for strongly convex functions when the solution lies in the boundary [16]. To overcome these limitations, variants of the FW algorithms with better convergence properties have been proposed. Two of these variants, the Away-steps FW [17] and Pairwise FW [18] enjoy a linear rate of convergence over polytopes [18].

Despite this theoretical breakthrough, Away-steps and Pairwise FW are not yet practical off-the-shelf solvers due to two main limitations. The first and most important is that both variants rely on an exact line-search. That is, they require to solve at each iteration 1-dimensional subproblems of the form

$$\underset{\gamma\in[0,\gamma_{\max}]}{\arg\min}\ f(\boldsymbol{x}_t+\gamma\boldsymbol{d}_t)\ , \tag{2.1}$$

where $\boldsymbol{d}_t$ is the update direction and $\gamma_{\max}$ is the maximum admissible step-size. In a few cases like quadratic objectives, the exact line-search subproblem has a closed form solution.

	Related work	non-convex analysis	approximate subproblems	linear convergence	adaptive step-size	bounded backtracking
Frank-Wolfe	This work	✓	✓	✓	✓	✓
	[18]	✗	✗	✓	✗	N/A
	[19]	✗	✓†	✗	✓	✗
	[20]	✓	✗	✗	✓	✗
MP	This work	✓	✓	✓	✓	✓
	[21]	✗	✓	✓	✗	N/A

Table 2.1: **Comparison with related work.** non-convex analysis: convergence guarantees for problems with a non-convex objective. approximate subproblems: convergence guarantees cover the case in which linear subproblems are solved approximately. linear convergence: guaranteed linear rate of convergence (under hypothesis). adaptive step-size: step-size is set using local information of the objective. bounded backtracking: explicit bound for the total number of inner iterations in adaptive step-size methods. † = assumes cartesian product domain.

In most other cases, it is a costly optimization problem that needs to be solved at each iteration, making these methods impractical. The second limitation is that they require access to an exact Linear Minimization Oracle (LMO), which leaves out important cases like minimization over a trace norm ball where the LMO is computed up to some predefined tolerance. In this paper we develop methods that lift both limitations simultaneously.

Our **main contribution** is the design and analysis of variants of Away-steps and Pairwise FW that *i*) don't require access to an exact line-search or knowledge of properties of the objective like its curvature or Lipschitz constant, and *ii*) admits the FW subproblems to be solved approximately. We describe our approach in §2.2. Although our main motivation is to develop practical variants of Away-steps and Pairwise FW, we also show that this technique extends to other methods like FW and Matching Pursuit.

We develop in §2.3 a convergence rate analysis for the proposed methods. The obtained rates match asymptotically the best known bounds on convex, strongly convex and non-convex problems, including linear convergence for strongly convex functions.

Finally, we show in §2.4 benchmarks between the proposed and related methods, and discuss the importance of large step-sizes in Pairwise FW.

Related work

We comment on the most closely related ideas, summarized in Table 2.1.

Away-Steps FW [17] is a popular variant of FW that adds the option to move away from an atom in the current representation of the iterate. In the case of a polytope domain, it was recently shown to enjoy a linear convergence rate for strongly convex objectives [22, 18]. Pairwise FW [18] simplifies the above-described variant by replacing the two kinds of steps by a single step modifying the weights of only two atoms. It generalizes the algorithm of

Mitchell, Demyanov, and Malozemov [23] used in geometry and SMO [24] for training SVMs. These methods all require exact line-search.

Variants of FW that, like the proposed methods, set the step-size based on a local decrease condition have been described by Dunn [20] and Beck, Pauwels, and Sabach [19], but none of these methods achieve a linear convergence rate to the best of our knowledge.

Matching Pursuit (MP) [25] is an algorithm for constrained optimization problems of the form Eqn. OPT with $\text{conv}(\boldsymbol{A})$ replaced by $\text{linspan}(\boldsymbol{A})$, the linear span of $\boldsymbol{A}$. Locatello et al. [26] has recently shown that MP and FW are deeply related. We show that our algorithm and convergence results also extend naturally to MP, and as a byproduct of our analysis we obtain the first convergence rate for MP on non-convex objectives to the best of our knowledge.

Notation. Throughout this chapter we denote vectors and vector-valued functions in lowercase boldface (e.g. $\boldsymbol{x}$ or $\arg\min$), matrices in uppercase boldface letters (e.g. $\boldsymbol{D}$), and sets in caligraphic letters (e.g., $\mathcal{A}$). We say a function f is L-smooth if it is differentiable and its gradient is L-Lipschitz continuous, that is, if it verifies $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain. A function is μ-strongly convex if $f - \frac{\mu}{2}\|\cdot\|^2$ is convex. $\|\cdot\|$ denotes the euclidean norm.

2.2 Methods

In this section we describe the core part of our contribution, which is a strategy to select the step-size in FW-type algorithms.

Since this strategy can be applied very broadly to Frank-Wolfe variants including Away-steps, Pairwise, classical FW and Matching Pursuit, we describe it within the context of a generic FW-like algorithm. This generic algorithm is detailed in Algorithm 1 and depends on two key functions: `update_direction` and `step_size`. The first one computes the direction that we will follow to compute the next iterate and its implementation will depend on the FW variant. The second one will choose an appropriate step-size based upon local information of the objective and is the key novelty of this algorithm. We now describe them in more detail.

Update direction

In this subsection we describe `update_direction` in Algorithm 1, the function that computes the update direction $\boldsymbol{d}_t$ and the maximum allowable step-size $\gamma_t^{\max}$. While its implementation varies according to the FW variant, all of them require to solve one or two linear problems, often referred to as linear minimization oracle (LMO).

The first of these subproblems is the same for all variants and consists in finding $\boldsymbol{s}_t$ in the domain such that:

$$\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle \leq \delta \min_{\boldsymbol{s} \in \mathcal{A}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} - \boldsymbol{x}_t \rangle \ . \tag{2.2}$$

Algorithm 1 Generic FW with backtracking

1: **Input:** $\boldsymbol{x}_0 \in \text{conv}(\mathcal{A})$, initial Lipschitz estimate $L_{-1} > 0$, tolerance $\varepsilon \geq 0$, subproblem quality $\delta \in (0, 1]$
2: **for** $t = 0, 1 \ldots$ **do**
3: $\quad \boldsymbol{d}_t, \gamma_t^{\max} = \text{update\_direction}(\boldsymbol{x}_t, \nabla f_t)$
4: $\quad g_t = \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle$
5: $\quad$ **if** $g_t \leq \delta\varepsilon$ **then**
6: $\quad\quad$ **return** $\boldsymbol{x}_t$
7: $\quad$ **end if**
8: $\quad \gamma_t, L_t = \text{step\_size}(f, \boldsymbol{d}_t, \boldsymbol{x}_t, g_t, L_{t-1}, \gamma_t^{\max})$
9: $\quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$
10: **end for**

Here, we introduce a subproblem quality parameter $\delta \in (0, 1]$ that allows this subproblem to be solved approximately. When $\delta = 1$, the problem is solved exactly and becomes $\arg\min_{\boldsymbol{s} \in \mathcal{A}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} \rangle$, which consists in selecting the atom that correlates the most with the steepest descent direction, $-\nabla f(\boldsymbol{x}_t)$.

Away-steps and Pairwise FW will also require to solve another linear subproblem, this time over the active set $\mathcal{S}_t$. This is the set of atoms with non-zero weight in the decomposition of $\boldsymbol{x}_t$. More formally, the active set $\mathcal{S}_t \subseteq \mathcal{A}$ is the set of atoms that have non-zero weight $\boldsymbol{\alpha}_{\boldsymbol{s},t} > 0$ in the expansion $\boldsymbol{x}_t = \sum_{\boldsymbol{s} \in \mathcal{S}_t} \boldsymbol{\alpha}_{\boldsymbol{v},t} \boldsymbol{s}$.

The linear subproblem that needs to be solved consists in finding $\boldsymbol{v}_t$ such that:

$$\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v}_t \rangle \leq \delta \min_{\boldsymbol{v} \in S_t} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v} \rangle \ . \tag{2.3}$$

Unlike the previous linear subproblem, this time the problem is over the typically much smaller active set $\mathcal{S}_t$. As before, $\delta \in (0, 1]$ allows this subproblem to be solved approximately. When $\delta = 1$, the subproblem becomes $\arg\max_{\boldsymbol{v} \in \mathcal{S}_t} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{v} \rangle$, which can be interpreted as selecting the atom in the active set that correlates the most with the steepest ascent direction $\nabla f(\boldsymbol{x}_t)$.

FW, AFW and PFW then combine the solution to these linear subproblems in different ways, and Line 3's `update_direction` is implemented as:

- FW returns $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{x}_t$ and $\gamma_t^{\max} = 1$: the next iterate will be a convex combination of $\boldsymbol{x}_t$ and $\boldsymbol{s}_t$.
- AFW considers directions $\boldsymbol{s}_t - \boldsymbol{x}_t$ and $\boldsymbol{x}_t - \boldsymbol{v}_t$, and chooses the one that correlates the most with $-\nabla f(\boldsymbol{x}_t)$. $\gamma_t^{\max} = 1$ if $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{x}_t$ and $\alpha_{\boldsymbol{v}_t}/(1 - \alpha_{\boldsymbol{v}_t})$ otherwise, where $\alpha_{\boldsymbol{v}_t}$ is the weight associated with $\boldsymbol{v}_t$ in the decomposition of $\boldsymbol{x}_t$ as a convex combination of atoms.
- PFW uses $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{v}_t$, shifting weight from $\boldsymbol{v}_t$ to $\boldsymbol{s}_t$ in our current iterate, and $\gamma_t^{\max} = \alpha_{\boldsymbol{v}_t}$.
- MP uses $\boldsymbol{d}_t = \boldsymbol{s}_t$ and $\gamma^{\max} = +\infty$, since the constraint set is not bounded.

Algorithm 2 Backtracking for FW variants

1: **Initialization:** f,$\boldsymbol{d}_t$, $\boldsymbol{x}_t$, g_t, L_{t-1}, $\gamma_{\max}$
2: Choose $\tau > 1$, $\eta \leq 1$
3: Choose $M \in [\eta L_{t-1}, L_{t-1}]$
4: $\gamma = \min\{g_t/(M\|\boldsymbol{d}_t\|^2), \gamma_{\max}\}$
5: **while** $f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t) > Q_t(\gamma, M)$ **do**
6: $\quad M = \tau M$
7: **end while**
8: $\gamma = \min\{g_t/(M\|\boldsymbol{d}_t\|^2), \gamma_{\max}\}$
9: **return** γ, M

Backtracking line-search

In this subsection we describe the step-size selection routine `step_size` (Algorithm 2). This is the main novelty in the proposed algorithms, and allows for the step-size to be computed using only local properties of the objective, as opposed to other approaches that use global quantities like the gradient's Lipschitz constant. As we will see in §2.4, this results in step-sizes that are often more than an order of magnitude larger than those estimated using global quantities.

Minimizing the exact line-search objective $\gamma \mapsto f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t)$ yields the highest decrease in objective but can be a costly optimization problem. To overcome this, we will replace the exact line-search objective with the following quadratic approximation:

$$Q_t(\gamma, M) = f(\boldsymbol{x}_t) - \gamma g_t + \frac{\gamma^2 M}{2}\|\boldsymbol{d}_t\|^2 \ . \tag{2.4}$$

This approximation has the advantage that its minimum over $\gamma \in [0, \gamma^{\max}]$ can be computed in closed form, which gives the step-size used in line 4:

$$\gamma_M^\star = \min\left\{\frac{g_t}{M\|\boldsymbol{d}_t\|^2}, \gamma^{\max}\right\} \ , \tag{2.5}$$

The quality of this quadratic approximation will depend on the Lipschitz estimate parameter M. This parameter needs to be carefully selected to maintain the convergence guaratees of exact line-search, while keeping the number of objective function evaluations to a minimum.

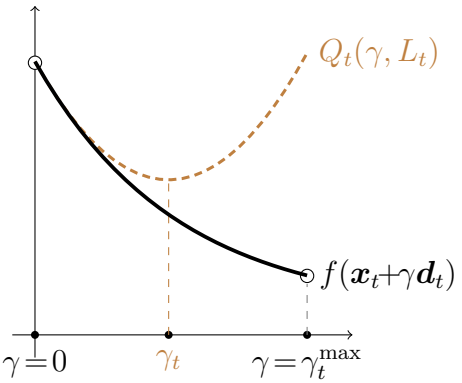
This is achieved through the strategy implemented in Algorithm 2. The algorithm initializes the Lipschitz estimate M to a value between ηL_{t-1} and the previous iterate L_{t-1}, where η is a user-defined parameter (default values discussed later). A value of $\eta = 1$ is admissible but would not allow the Lipschitz estimate to decrease through the optimization, and we have observed empirically a drastic benefit in doing so.

The algorithm then defines a candidate step-size γ (Line 4) and checks whether the following sufficient decrease condition is verified for this step-size

$$f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t) \leq Q_t(\gamma, M) \ , \gamma = \min\left\{g_t/(M\|\boldsymbol{d}_t\|^2), \gamma^{\max}\right\}.$$

If it is not verified, we increase this constant by a power factor of $\tau > 1$ (Line 6). By the properties of L-smooth functions, we know that this condition is verified for all $M \geq L$, and so this loop has a finite termination.

Once this condition is verified, the current step-size is accepted and the value of M is assigned the name L_t. Geometrically, the sufficient decrease condition ensures that the quadratic approximation is an upper bound at its constrained minimum of the line-search objective. We emphasize that this upper bound does not need to be a global one, as it only holds at γ_t. This allows for smaller L_t than the global Lipschitz constant L, and therefore larger step-sizes.



As we will see in §2.3, this translates into faster convergence rates that depend on L_t, as well as faster empirical convergence (§2.4).

Default and initial parameters. Algorithm 1 requires an (arbitrary) initial value for the Lipschitz estimate L_{-1}. We found the following heuristic using the definition of Lipschitz continuity of the gradient to work well in practice. Select a small constant ε, say 10^{-3}, and compute $L_{-1} = \|\nabla f(\boldsymbol{x}_0) - \nabla f(\boldsymbol{x}_0 + \varepsilon \boldsymbol{d}_0)\|/(\varepsilon \|\boldsymbol{d}_0\|)$.

The `step_size` depends on hyperparameters η and τ. Although the algorithm is guaranteed to converge for any $\eta \leq 1$, $\tau > 1$, we recommend $\eta = 0.9, \tau = 2$, as we found that it performs well in a variety of scenarios. These are the values used throughout benchmarks §2.4.

This method also requires to choose the initial value of the Lipschitz estimate M to a value between ηL_{t-1} and L_{t-1}. A choice that we found to work remarkably well in practice is to initialize it to

$$M = \text{clip}_{[\eta L_{t-1}, L_{t-1}]} \left(\frac{g_t^2}{2(f_{t-1} - f_t)\|\boldsymbol{d}_t\|^2} \right) . \tag{2.6}$$

The value inside the clip function corresponds to the optimal value of M for a quadratic interpolation between the previous two iterates and the derivative of the line-search objective $f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t)$ at $\gamma = 0$. Since this value might be outside of the interval $[\eta L_{t-1}, L_{t-1}]$, we clip the result to this interval.

Pseudocode and implementation details. A practical implementation of these algorithms depends on other details that are not specific to the backtracking variant, such as efficiently maintaining the active-set in the case of Away-steps and Pairwise. For completeness, B.1 contains a full pseudocode for all these algorithms. A Python implementation of these methods, as well as the benchmarks used in §2.4 will be made open source upon publication of this manuscript.

2.3 Analysis

In this section, we provide a convergence rate analysis of the proposed methods, showing that all enjoy a $\mathcal{O}(1/\sqrt{t})$ convergence rate for non-convex objectives (Theorem 2), a stronger $\mathcal{O}(1/t)$ convergence rate for convex objectives (Theorem 3), and for some variants linear convergence for strongly convex objectives over polytopes (Theorem 4).

Notation. In this section we make use of the following extra notation:

- For convenience we will refer to the variants of FW, Away-steps FW, Pairwise FW and MP with backtracking line-search as AdaFW, AdaAFW, AdaPFW and AdaMP respectively.
- We denote the objective suboptimality at step t as $h_t = f(\boldsymbol{x}_t) - \min_{\boldsymbol{x}\in\text{conv}(\mathcal{A})} f(\boldsymbol{x})$.
- Good and bad steps. Our analysis, as that of Lacoste-Julien and Jaggi [18], relies on a notion of "good" and "bad" steps. We define bad steps as those that verify $\gamma_t = \gamma_t^{\max}$ and $\gamma_t^{\max} < 1$ and good steps as any step that is not a bad step. The name "bad steps" makes reference to the fact that we won't be able to bound non-trivially the improvement for these steps. For these steps we will only be able to guarantee that the objective is non-increasing. AdaAFW and AdaPFW both may have bad steps. Let us denote by N_t the number of "good steps" up to iteration t. We can lower bound the number of good steps by

$$N_t \geq t/2 \text{ for AdaAFW}\,, \tag{2.7}$$
$$N_t \geq t/(3|\mathcal{A}|! + 1) \text{ for AdaPFW} \tag{2.8}$$

  where it is worth noting that the last bound for AdaPFW requires the set of atoms $\mathcal{A}$ to be finite. The proof of these bounds can be found in B.3 and are a direct translation of those in [18]. We have found these bounds to be very loose, as in practice the fraction of bad/good steps is negligible, commonly of the order of 10^{-5} (see last column of the table in Figure 2.1).

- Average and maximum of Lipschitz estimates. In order to highlight the better convergence rates that can be obtained by adaptive methods we introduce the average and maximum estimate over good step-sizes. Let $\mathcal{G}_t$ denote the indices of good steps up to iteration t. Then we define the average and maximum Lipschitz estimate as

$$\overline{L}_t \stackrel{\text{def}}{=} \frac{1}{N_t}\sum_{k\in\mathcal{G}_t} L_k \tag{2.9}$$
$$L_t^{\max} \stackrel{\text{def}}{=} \max_{k\in\mathcal{G}_t} L_k \tag{2.10}$$

  respectively. In the worst case, both quantities can be upper bounded by $\max\{\tau L, L_{-1}\}$ (Proposition 4), which can be used to obtain asymptotic convergence rates. This bound is however very pessimistic. We have found that in practice $\overline{L}_t$ is often more than 100 times smaller than L (see second to last column of the table in Figure 2.1).

Algorithm	Non-convex	Convex	Strongly convex
AdaAFW	$\mathcal{O}(\frac{1}{\delta\sqrt{t}})$	$\mathcal{O}(\frac{1}{\delta^2 t})$	$\mathcal{O}((1-\delta^2\rho)^t)$
AdaPFW	$\mathcal{O}(\frac{1}{\delta\sqrt{t}})$	$\mathcal{O}(\frac{1}{\delta^2 t})$	$\mathcal{O}((1-\delta^2\rho)^t)$
AdaFW	$\mathcal{O}(\frac{1}{\delta\sqrt{t}})$	$\mathcal{O}(\frac{1}{\delta^2 t})$	$\mathcal{O}(\frac{1}{\delta^2 t})$
AdaMP	$\mathcal{O}(\frac{1}{\delta\sqrt{t}})$	$\mathcal{O}(\frac{1}{\delta^2 t})$	$\mathcal{O}((1-\delta^2\rho_{\text{MP}})^t)$

Table 2.2: **Convergence rate summary** on non-convex, convex and strongly convex objectives. For non-convex objectives, bound is on the minimum FW gap (MP gap in the case of AdaMP), in other cases its on the objective suboptimality.

Our new convergence rates are presented in the following theorems, which consider the cases of non-convex, convex and strongly convex objectives. The results are discussed in §2.3 and the proofs can be found in B.4, B.5 and B.6 respectively.

Overhead of backtracking

Evaluation of the sufficient decrease condition Algorithm 2 requires two extra evaluations of the objective function. If the condition is verified, then it is only evaluated at the current and next iterate. FW requires anyway to compute the gradient at these iterates, hence in cases in which the objective function is available as a byproduct of the gradient this overhead becomes negligible.

Furthermore, we can provide a bound on the total number of evaluations of the sufficient decrease condition:

Theorem 1. *Let n_t be the total number of evaluations of the sufficient decrease condition up to iteration t. Then we have*

$$n_t \leq \left[1 - \frac{\log \eta}{\log \tau}\right](t+1) + \frac{1}{\log \tau} \max\left\{\log \frac{\tau L}{L_{-1}}, 0\right\} ,$$

This result highlights the trade-off faced when choosing η. Minimizing it with respect to η gives $\eta = 1$, in which case $(1 - \log \eta / \log \tau) = 1$ and so there's an asymptotically vanishing number of failures in the sufficient decrease condition. Unfortunately, $\eta = 1$ also forbids the Lipschitz estimate to decrease along the optimization. Ideally, we would like η small enough so that the Lipschitz estimate decreases when it can, but not too small so that we waste too much time in failed sufficient decrease evaluations.

As mentioned before, we recommend parameters $\eta = 0.9$, $\tau = 2$. With these values, we have that $\left[1 - \frac{\log \eta}{\log \tau}\right] \leq 1.16$, and so asymptotically no more than 16% of the iterates will result in more than one evaluations of the sufficient decrease condition.

Non-convex objectives

Gap function. Convergence rates for convex and strongly convex functions are given in terms of the objective function suboptimality or a primal-dual gap. As the gap upper-bounds (i.e. certifies) the suboptimality, the latter is a stronger result in this scenario. In the case of non-convex objectives, as is common for first order methods, we will only be able to guarantee convergence to a stationary point, defined as any element $\boldsymbol{x}^\star \in \operatorname{conv}(\mathcal{A})$ such that $\langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x} - \boldsymbol{x}^\star \rangle \geq 0$ for all $\boldsymbol{x} \in \operatorname{conv}(\mathcal{A})$ [27].

Following Lacoste-Julien [28] and Reddi et al. [29], for FW variants we will use as convergence criterion the FW gap, defined as $g^{\mathrm{FW}}(\boldsymbol{x}) = \max_{\boldsymbol{s} \in \operatorname{conv}(\mathcal{A})} \langle \nabla f(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{s} \rangle$. From the definition of stationary point it is clear that the FW gap is zero only at a stationary point. These rates are also valid for AdaMP, albeit for the more appropriate gap function g^{MP} detailed in B.4.

Theorem 2. *Let $\boldsymbol{x}_t$ denote the iterate generated by any of the proposed algorithms after t iterations, with $N_{t+1} \geq 1$. Then we have:*

$$\lim_{t \to \infty} g(\boldsymbol{x}_t) = 0 \qquad \textit{and} \tag{2.11}$$

$$\min_{k=0,\dots,t} g(\boldsymbol{x}_k) \leq \frac{C_t}{\delta \sqrt{N_{t+1}}} = \mathcal{O}\left(\frac{1}{\delta \sqrt{t}}\right), \tag{2.12}$$

where $C_t = \max\{2h_0, L_t^{\max} \operatorname{diam}(\boldsymbol{A})^2\}$ and $g = g^{FW}$ is the FW gap for AdaFW, AdaAFW, AdaPFW and $C_t = \operatorname{radius}(\mathcal{A}) \sqrt{2h_0 \overline{L}_{t+1}}$ and $g = g^{MP}$ is the MP gap for AdaMP.

Convex objectives

For convex objectives we will be able to improve the results of Theorem 2. We will first state the convergence results for FW variants and then for MP.

For adaptive FW variants, we will be able to give an $\mathcal{O}(1/\delta^2 t)$ convergence rate on the primal-dual gap, which trivially implies a bound on the objective suboptimality. In order to define the primal-dual gap, we define the following dual objective function

$$\psi(\boldsymbol{u}) \stackrel{\text{def}}{=} -f^*(\boldsymbol{u}) - \sigma_{\operatorname{conv}(\mathcal{A})}(-\boldsymbol{u}) \;, \tag{2.13}$$

where f^* denotes the convex conjugate of f and $\sigma_{\operatorname{conv}(\mathcal{A})}(\boldsymbol{x}) \stackrel{\text{def}}{=} \sup\{\langle \boldsymbol{x}, \boldsymbol{a} \rangle : \boldsymbol{a} \in \operatorname{conv}(\mathcal{A})\}$ is the support function over $\operatorname{conv}(\mathcal{A})$, which is the convex conjugate of the indicator function. Note that ψ is concave and that when f convex, we have by duality $\min_{\boldsymbol{x} \in \operatorname{conv}(\mathcal{A})} f(\boldsymbol{x}_t) = \max_{\boldsymbol{u} \in \mathbb{R}^p} \psi(\boldsymbol{u})$.

Theorem 3 (FW variants)**.** *Let f be convex, $\boldsymbol{x}_t$ denote the iterate generated by any of the proposed FW variants (AdaFW, AdaAFW, AdaPFW) after t iterations, with $N_t \geq 1$, and let $\boldsymbol{u}_t$ be defined recursively as $\boldsymbol{u}_0 = \nabla f(\boldsymbol{x}_0)$, $\boldsymbol{u}_{t+1} = (1 - \xi_t)\boldsymbol{u}_t + \xi_t \nabla f(\boldsymbol{x}_t)$, where*

$\xi_t = 2/(\delta N_t + 2)$ *if* t *is a good step and* $\xi_t = 0$ *otherwise. Then we have:*

$$h_t \leq f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t) \tag{2.14}$$

$$\leq \frac{2\overline{L}_t \operatorname{diam}(\boldsymbol{A})^2}{\delta^2 N_t + \delta} + \frac{2(1-\delta)}{\delta^2 N_t^2 + \delta N_t}\big(f(\boldsymbol{x}_0) - \psi(\boldsymbol{u}_0)\big)$$

$$= \mathcal{O}\left(\frac{1}{\delta^2 t}\right). \tag{2.15}$$

Strongly convex objectives

The next result states the linear convergence of some algorithm variants and uses the notions of pyramidal width (PWidth) and minimal directional width (mDW) that have been developed in [28] and [21] respectively, which we state in B.2 for completeness. We note that the pyramidal width of a set $\boldsymbol{A}$ is lower bounded by the minimal width over all subsets of atoms, and thus is strictly greater than zero if the number of atoms is finite. The minimal directional width is a much simpler quantity and always strictly greater than zero by the symmetry of our domain.

Theorem 4 (Linear convergence rate for strongly convex objectives)**.** *Let* f *be* μ*–strongly convex. Then for AdaAFW, AdaPFW or AdaMP we have the following linear decrease for each good step* t*:*

$$h_{t+1} \leq (1 - \delta^2 \rho_t) h_t \,, \tag{2.16}$$

where

$$\rho_t = \frac{\mu}{4L_t}\left(\frac{\operatorname{PWidth}(\boldsymbol{A})}{\operatorname{diam}(\boldsymbol{A})}\right)^2 \text{ for AdaAFW and AdaPFW,}$$

$$\rho_t = \frac{\mu}{L_t}\left(\frac{\operatorname{mDW}(\boldsymbol{A})}{\operatorname{radius}(\boldsymbol{A})}\right)^2 \text{ for AdaMP.}$$

The previous theorem gives a geometric decrease on good steps. Combining this theorem with the bound for the number of bad steps in Eqn. 2.7, and noting that the sufficient decrease guarantees that the objective is monotonically decreasing, we obtain a global linear convergence for AdaAFW, AdaPFW and AdaMP.

Discussion

Non-convex objectives. Lacoste-Julien [28] studied the convergence of FW assuming the linear subproblems are solved exactly ($\delta = 1$) and obtained a rate of the form Eqn. 2.11 with $C_0 = \max\{2h_0, L \operatorname{diam}(\operatorname{conv}(\mathcal{A}))^2\}$ instead. Both rates are similar, although our analysis is more general as it allows to consider the case in which linear subproblems are solved approximately ($\delta < 1$) and also gives rates for the Away-steps and Pairwise variants, for which no rates were previously known.

Theorem 2 also gives the first known convergence rates for a variant of MP on general non-convex functions. Contrary to the case of FW, this bound depends on the mean instead of the maximum of the Lipschitz estimate.

Convex objectives. Compared with [15], the primal-dual rates of Theorem 3 are stronger as they hold for the last iterate and not only for the minimum over previous iterates. To the best of our knowledge, primal-dual convergence rates on the last iterate have only been derived in [30] and were not extended to approximate linear subproblems nor the Away-steps and Pairwise variants.

Compared to Nesterov [30] on the special case of exact subproblems ($\delta = 1$), the rates of Theorem 3 are similar but with $\overline{L}_t$ replaced by L. Hence, in the regime $\overline{L}_t \ll L$ (as is often verified in practice), our bounds have a much smaller leading constant.

For MP, Locatello et al. [26] obtain a similar convergence rate of the form $\mathcal{O}(1/(\delta^2 t))$, but with constants that depend on global properties of ∇f, instead of the adaptive, averaged Lipschitz estimate in our case.

Strongly convex objectives. For the FW variants, the rates are identical to the ones in [18, Theorem 1], with the significant difference of replacing L with the adaptive L_t in the linear rate factor, giving a larger per-iteration decrease whenever $L_t < L$. Our rates are the first also covering approximate subproblems for Away-Steps and Pairwise FW algorithms. It's also worth noticing that both Away-steps FW and Pairwise FW have only been previously analyzed in the presence of exact line-search [18]. Additionally, unlike [18], we do not require a smoothness assumption on f outside of the domain. Finally, for the case of MP, we again obtain the same convergence rates as in [21, Theorem 7], but with L replaced by L_t.

2.4 Benchmarks

We compared the proposed methods across three problems and three datasets. The three datasets are summarized in the table of Figure 2.1, where density denotes the fraction of nonzero coefficients in data matrix and where the last two columns are quantities that arise during the optimization of AdaPFW and shed light into their empirical value. In both cases t is the number of iterates until 10^{-10} suboptimality is achieved.

ℓ_1-constrained logistic regression

The first problem that we consider is a logistic regression with an ℓ_1 norm constraint on the coefficients of the form:

$$\underset{\|\boldsymbol{x}\|_1 \leq \beta}{\arg\min} \ \frac{1}{n} \sum_{i=1}^{n} \varphi(\boldsymbol{a}_i^\top \boldsymbol{x}, \boldsymbol{b}_i) + \frac{\lambda}{2} \|\boldsymbol{x}\|_2^2 \ , \tag{2.17}$$

where φ is the logistic loss. β is chosen to give approximately 1%, 20% of nonzero coefficients respectively. The linear subproblems in this case can be computed exactly ($\delta = 1$) and consist of finding the largest entry of the gradient. The ℓ_2 regularization parameter λ is always set to $\lambda = \frac{1}{n}$.

Dataset	#samples	#features	density	$\overline{L}_t/L$	$(t-N_t)/t$
Madelon [31]	4400	500	1.	3.3×10^{-3}	5.0×10^{-5}
RCV1 [32]	697641	47236	10^{-3}	1.3×10^{-2}	7.5×10^{-5}
MovieLens 1M [33]	6041	3707	0.04	1.1×10^{-2}	–

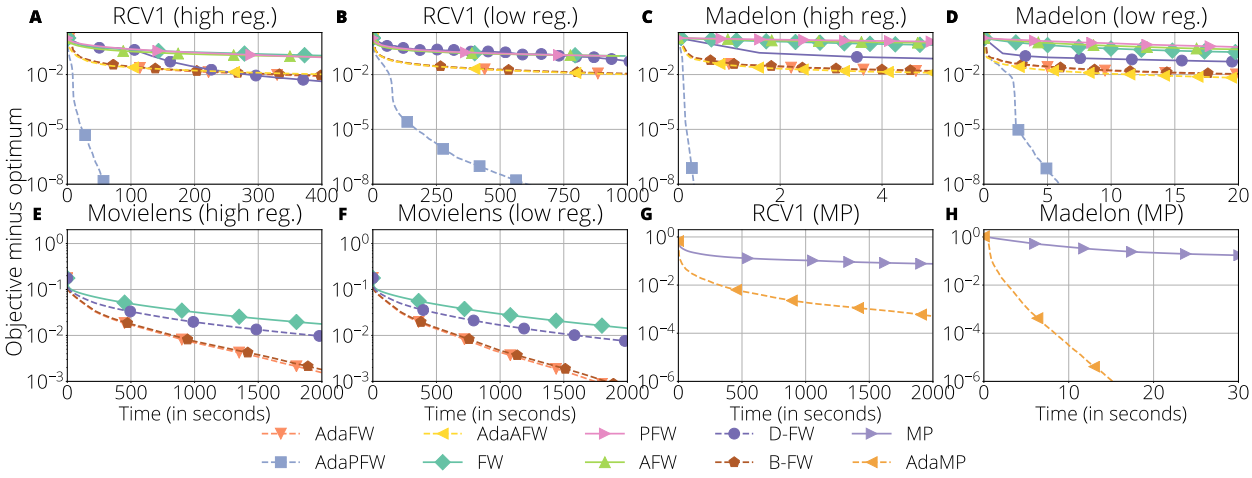


Figure 2.1: **Top table**: description of the datasets. **Bottom figure**: Benchmark of different FW and MP variants. The variants with backtracking line-search proposed in this paper are in dashed lines. Problem in A, B, C, D = logistic regression with ℓ_1-constrained coefficients, in E, F = Huber regression with on the nuclear norm constrained coefficients and in G, H = unconstrained logistic regression (MP variants). In all the considered datasets and regularization regimes, backtracking variants have a much faster convergence than others.

We applied this problem on two different datasets: Madelon and RCV1. We show the results in Figure 2.1, subplots A, B, C, D. In this figure we also show the performance of FW, Away-steps FW (AFW) and Pairwise FW (PFW), all of them using the step-size $\gamma_t = \min\{g_t L^{-1} \|\boldsymbol{d}_t\|^{-2}, \gamma_t^{\max}\}$, as well as the backtracking variants of Dunn [20] and [19], which we denote D-FW and B-FW respectively.

Nuclear-norm constrained Huber regression

The second problem that we consider is collaborative filtering. We used the MovieLens 1M dataset, which contains 1 million movie ratings, and consider the problem of minimizing a Huber loss, as in [34], between the true known ratings and a matrix $\boldsymbol{X}$. We also constrain the matrix by its nuclear norm $\|\boldsymbol{X}\|_* \leq \beta$, where β is chosen to give approximately 1% and 20% of non-zero singular values respectively. The problem is of the form:

$$\underset{\|\boldsymbol{X}\|_* \leq \beta}{\arg\min} \frac{1}{n} \sum_{(i,j)\in\mathcal{I}}^{n} L_\xi(\boldsymbol{A}_{i,j} - \boldsymbol{X}_{i,j}) \ , \tag{2.18}$$

where H_1 is the Huber loss, defined as

$$L_\xi(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \xi, \\ \xi(|a| - \frac{1}{2}\xi), & \text{otherwise}\,. \end{cases}$$

The Huber loss is a quadratic for $|a| \leq \xi$ and grows linearly for $|a| > \xi$. The parameter ξ controls this tradeoff and was set to 1 during the experiments.

In this case, the AFW and PFW variants were not considered as they are not directly applicable to this problem as the size of the active set is potentially unbounded. The results of this comparison can be see in subplots E and F of Figure 2.1. We emphasize that the goal of this experiment is to compare different FW variants and not find the best method for matrix completion. For alternative approaches not based on FW see for instance [35].

We comment on some observed trends from these results:

- **Importance of backtracking.** Across the different datasets, problems and regularization regimes we found that backtracking methods always perform better than their non-backtracking variant.

- **Pairwise FW.** AdaPFW shows a surprisingly good performance when it is applicable, specially in the high regularization regime. A possible interpretation for this is that it is the only variant of FW in which the coefficients associated with previous atoms are not shrunk when adding a new atom, hence large step-sizes are potentially even more beneficial as coefficients that are already close to optimal do not get necessarily modified in subsequent updates.

- $\overline{L}_t$ **vs** L**.** We compared the average Lipschitz estimate $\overline{L}_t$ and the L, the the gradient's Lipschitz constant. We found that across all datasets the former was more than an order of magnitude smaller, highlighting the need to use a local estimate of the Lipschitz constant to use a large step-size.

- **Bad steps.** Despite the very pessimistic bounds obtained for the number of bad steps in the previous section, we observe that in practice these are extremely rare events, happening less than once every 10,000 iterations.

2.5 Conclusion and Future Work

In this work we have proposed and analyzed a novel adaptive step-size scheme that can be used in projection-free methods such as FW and MP. The method has minimal computational overhead and does not rely on any step-size hyperparameter (except for an initial estimate). Numerical experiments show large computational gains on a variety of problems.

A possible extension of this work is to develop backtracking step-size strategies for randomized variants of FW such as [36, 37, 38], in which there is stochasticity in the linear subproblems.

Another area of future research is to improve the convergence rate of the Pairwise FW method. Due to the very pessimistic bound on its number of bad steps, there is still a large gap between its excellent empirical performance and its known convergence rate. Furthermore, convergence of Pairwise and Away-steps for an infinite $\mathcal{A}$, such as the trace norm ball, is still an open problem.

Acknowledgements

The authors would like to thank Vlad Niculae and Courtney Paquette for valuable feedback on the manuscript.

Chapter 3

Stochastic Frank-Wolfe for constrained finite-sum minimization

3.1 Introduction

We consider constrained finite-sum optimization problems of the form

$$\underset{\boldsymbol{w}\in\mathcal{C}}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n} f_i\left(\boldsymbol{x}_i^\top \boldsymbol{w}\right), \tag{OPT}$$

where $\mathcal{C}$ is a compact and convex set and $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\top \in \mathbb{R}^{n\times d}$ is a data matrix, with n samples and d features. This template includes several problems of interest, such as constrained empirical risk minimization. The LASSO [39] may be written in this form, where $f_i(\boldsymbol{x}_i^\top \boldsymbol{w}) = \frac{1}{2}(\boldsymbol{x}_i^\top \boldsymbol{w} - y_i)^2$ and $\mathcal{C} = \{\boldsymbol{w} : \|\boldsymbol{w}\|_1 \leq \lambda\}$ for some parameter λ. We focus on the case where the f_is are differentiable with L-Lipschitz derivative, and study the convex and non-convex cases.

The classical Frank-Wolfe (FW) or Conditional Gradient algorithm [12, 13, 14] is an algorithm for constrained optimization. Contrary to other projection-based constrained optimization algorithms, such as Projected Gradient Descent, it relies on a Linear Minimization Oracle (LMO) over the constraint set $\mathcal{C}$, rather than a Quadratic Minimization Oracle (the projection subroutine). For certain constraint sets such as the trace norm or most ℓ_p balls, the LMO can be computed more efficiently than the projection subroutine. Recently, the Frank-Wolfe algorithm has garnered much attention in the machine learning community where polytope constraints and sparsity are of large interest, e.g. Jaggi [15], Lacoste-Julien and Jaggi [18], and Locatello et al. [21].

In the unconstrained setting, stochastic variance-reduced methods [40, 41, 42] exhibit the same iteration complexity as full gradient (non-stochastic) methods, while reaching much smaller per-iteration complexity, usually at some (small) additional memory cost. This work takes a step in the direction of designing such a method for Frank-Wolfe type algorithms, which remains an important open problem.

Table 3.1: Worst-case convergence rates for the function suboptimality after t iterations, for a dataset with n samples. $\kappa \leq n$ and can be much smaller than n for datasets of interest. κ is introduced in Section 3.5.

Related Work	Convex	Non-Convex
Frank and Wolfe [12]	$\mathcal{O}(n/t)$	$\mathcal{O}(n/\sqrt{t})$
Mokhtari, Hassani, and Karbasi [38]	$\mathcal{O}(1/\sqrt[3]{t})$	✗
Lu and Freund [43]	$\mathcal{O}(n/t)$	✗
This work	$\mathcal{O}(\kappa/t)$	$\to 0$

The main contributions of this chapter are:

1. A **constant batch-size Stochastic Frank-Wolfe (SFW) algorithm for finite sums with linear prediction**. We describe the method in Section 3.2 and discuss its computational and memory cost.

2. A **non-asymptotic rate analysis on smooth and convex objectives**. The suboptimality of the SFW algorithm after t iterations can be bounded as $\mathcal{O}(\kappa/t)$, where κ is a data-dependent constant we will discuss later. It is upper bounded by the sample-size n but, depending on the setting, can be potentially much smaller.

3. An **asymptotic analysis for non-convex objectives**. We prove that SFW converges to a stationary point for smooth but potentially non-convex functions. This is the first stochastic FW variant that has convergence guarantees in this setting of large practical interest.

Finally, we compare the SFW algorithm with other stochastic Frank-Wolfe algorithms amenable to constant batch size on different machine learning tasks. These experiments show that the proposed method converges at least as fast as previous work, and notably faster on several such instances.

Related Work

We split existing stochastic FW algorithm into two categories: methods with increasing batch size and methods with constant batch size.

Increasing batch size Stochastic Frank-Wolfe. This variant allows the number of gradient evaluations to grow with the iteration number [44, 45, 29]. Because of the growing number of gradient evaluations, these methods converge towards a deterministic full gradient FW algorithm and so asymptotically share their computational requirements. In this work we will instead be interested in constant batch-size methods, in which the number of gradient evaluations does not increase with the iteration number. See Hazan and Luo [45] for a

detailed comparison of assumptions and complexities for Stochastic Frank-Wolfe methods with increasing batch sizes, in terms of both iterations and gradient calls.

Constant batch size Stochastic Frank-Wolfe. These methods use a constant batch size b, which is chosen by the user as a hyperparameter. In the convex and smooth setting, Mokhtari, Hassani, and Karbasi [38] and Locatello et al. [46] reach $\mathcal{O}\left(1/\sqrt[3]{t}\right)$ convergence rates. The rate of Locatello et al. [46] further holds for non-smooth and non-Lipschitz objectives. Zhang et al. [47] requires second order knowledge of the objective. Lu and Freund [43] proves convergence for an averaged iterate in $\mathcal{O}(n/t)$ with n the number of samples in the dataset. Let us assume for simplicity that we use unit batch size. Since each iteration involves only one data point, the per-iteration complexity of their method reduces by a factor of n the per-iteration complexity of full-gradient method. On the other hand, the method proposed in this work loses this factor in the rate in number of iterations, reaching the same overall complexity as the deterministic full gradient method. Depending on the use-case (large or small datasets), each of the rates reported in Lu and Freund [43] and Mokhtari, Hassani, and Karbasi [38] can have an advantage over the other. In favorable cases, the rate of convergence achieved by our method is nearly independent of the number of samples in the dataset. In these cases, our method is therefore faster than both. In the worst case, it matches the $\mathcal{O}\left(n/t\right)$ bound [43].

Notation

In this chapter, we denote vectors in lowercase boldface letters ($\boldsymbol{w}$), matrices in uppercase boldface letters ($\boldsymbol{X}$), and sets in calligraphic letters (e.g., $\mathcal{C}$). We say a function f is L-smooth in the norm $\|\cdot\|$ if it is differentiable and its gradient is L-Lipschitz continuous with respect to $\|\cdot\|$, that is, if it verifies $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_* \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain (where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$). For a one dimensional function f, this reduces to $|f'(z) - f'(z')| \leq L|z - z'|$ for all $z,\ z'$ in the domain. For the time dependent vector $\boldsymbol{u}_t$, we denote by $\boldsymbol{u}_t^{(i)}$ its i-th coordinate.

We distinguish $\mathbb{E}$, the full expectation taken with respect to all the randomness in the system, from $\mathbb{E}_t$, the conditional expectation with respect to the random index sampled at iteration t, conditioned on all randomness up to iteration t.

Finally, $\mathrm{LMO}(\boldsymbol{u})$ returns an arbitrary element in $\arg\min_{\boldsymbol{s}\in\mathcal{C}}\langle \boldsymbol{s}, \boldsymbol{u}\rangle$.

3.2 Methods

A Primal-Dual View on Frank-Wolfe

In this subsection, we present the Frank-Wolfe algorithm as an alternating optimization scheme on a saddle-point problem. This point of view motivates the design of the proposed SFW algorithm. This perspective is similar to the two player game point of view of Abernethy

and Wang [48] and Abernethy et al. [49], which we express using convex conjugacy. We suppose here that f is closed, convex and differentiable.

Let us rewrite our initial problem Eqn. OPT in the equivalent unconstrained formulation

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{X}\boldsymbol{w}) + \imath_{\mathcal{C}}(\boldsymbol{w}) \;, \tag{3.1}$$

where $\imath_{\mathcal{C}}$ is the indicator function of $\mathcal{C}$: it is 0 over $\mathcal{C}$ and $+\infty$ outside of $\mathcal{C}$.

We denote by f^* the convex conjugate of f, that is, $f^*(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \max_{\boldsymbol{w}} \langle \boldsymbol{\alpha}, \boldsymbol{w}\rangle - f(\boldsymbol{w})$. Whenever f is closed and convex, it is known that $f = (f^*)^*$, and so we can write $f(\boldsymbol{X}\boldsymbol{w}) = \max_{\boldsymbol{\alpha}}\{-f^*(\boldsymbol{\alpha}) + \langle \boldsymbol{X}\boldsymbol{w}, \boldsymbol{\alpha}\rangle\}$. Plugging this identity into the previous equation, we arrive at a saddle-point reformulation of the original problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \left\{ \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) \stackrel{\text{def}}{=} -f^*(\boldsymbol{\alpha}) + \imath_{\mathcal{C}}(\boldsymbol{w}) + \langle \boldsymbol{X}\boldsymbol{w}, \boldsymbol{\alpha}\rangle \right\}. \tag{3.2}$$

This reformulation allows to derive the Frank-Wolfe algorithm as an alternating optimization method on this saddle-point reformulation. To distinguish the algorithm in this section from the stochastic algorithm we propose, we denote the iterates in this section by $\bar{\boldsymbol{\alpha}}_t$, $\bar{\boldsymbol{w}}_t$.

The first step of the Frank-Wolfe algorithm is to compute the gradient of the objective at the current iterate. In the saddle-point formulation, this corresponds to maximizing over the dual variable $\boldsymbol{\alpha}$ at step t:

$$\begin{aligned} \bar{\boldsymbol{\alpha}}_t &\in \arg\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \{\mathcal{L}(\bar{\boldsymbol{w}}_{t-1}, \boldsymbol{\alpha}) = -f^*(\boldsymbol{\alpha}) + \langle \boldsymbol{X}\bar{\boldsymbol{w}}_{t-1}, \boldsymbol{\alpha}\rangle\} \\ &\iff \bar{\boldsymbol{\alpha}}_t = \nabla f(\boldsymbol{X}\bar{\boldsymbol{w}}_{t-1}). \end{aligned} \tag{3.3}$$

Then, the LMO step corresponds to fixing the dual variable and minimizing over the primal one $\boldsymbol{w}$. This gives

$$\begin{aligned} \bar{\boldsymbol{s}}_t &\in \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \left\{\mathcal{L}(\boldsymbol{w}, \bar{\boldsymbol{\alpha}}_t) = \imath_{\mathcal{C}}(\boldsymbol{w}) + \langle \boldsymbol{w}, \boldsymbol{X}^\top \bar{\boldsymbol{\alpha}}_t\rangle\right\} \\ &\iff \bar{\boldsymbol{s}}_t = \mathrm{LMO}(\boldsymbol{X}^\top \bar{\boldsymbol{\alpha}}_t). \end{aligned} \tag{3.4}$$

Note that from the definition of the LMO, $\bar{\boldsymbol{s}}_t$ can always be chosen as an extreme point of the constraint set $\mathcal{C}$. We then update our iterate using the convex combination

$$\bar{\boldsymbol{w}}_t = (1-\gamma_t)\bar{\boldsymbol{w}}_{t-1} + \gamma_t \bar{\boldsymbol{s}}_t, \tag{3.5}$$

where γ_t is a step-size to be chosen. These updates determine the Frank-Wolfe algorithm.

The Stochastic Frank-Wolfe Algorithm

We now consider a variant in which we replace the exact minimization of the dual variable Eqn. 3.3 by a minimization over a single coordinate, chosen uniformly at random.

Let us define the function f from $\mathbb{R}^n$ to $\mathbb{R}$ as $f(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{\theta}_i)$. We can write our original optimization problem as an optimization over $\boldsymbol{w} \in \mathcal{C}$ of $f(\boldsymbol{X}\boldsymbol{w})$. Still alternating

Algorithm 3 Stochastic Frank-Wolfe

Initialization: $\boldsymbol{w}_0 \in \mathcal{C}$, $\boldsymbol{\alpha}_0 \in \mathbb{R}^n$, $\mathrm{r}_0 = \boldsymbol{X}^\top \boldsymbol{\alpha}_0$
for $t = 1, 2, \ldots,$ **do**
  Sample $i \in \{1, \ldots, n\}$ uniformly at random.
  Update $\boldsymbol{\alpha}_t^{(i)} = \frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$
  Update $\boldsymbol{\alpha}_t^{(j)} = \boldsymbol{\alpha}_{t-1}^j$, $j \neq i$
  $\mathrm{r}_t = \mathrm{r}_{t-1} + (\boldsymbol{\alpha}_t^{(i)} - \boldsymbol{\alpha}_{t-1}^{(i)})\boldsymbol{x}_i$
  $\boldsymbol{s}_t = \mathrm{LMO}(\mathrm{r}_t)$
  $\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \frac{2}{t+2}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})$
end for

between the primal and the dual problems, we replace maximization over the full vector $\boldsymbol{\alpha}$ in Eqn. 3.3 with optimization along the coordinate i only. We obtain the update $\boldsymbol{\alpha}_t^{(i)} = \frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$. Doing so changes the cost per-iteration from $\mathcal{O}(nd)$ to $\mathcal{O}(d)$, and yields Algorithm 3.

We now describe our main contribution, Algorithm 3 (SFW) above. It follows the classical Frank-Wolfe algorithm, but replaces the gradient with a stochastic estimate of the gradient.

Throughout Algorithm 3, we maintain the following iterates:

- the iterate $\boldsymbol{w}_t$,
- the stochastic estimator of $\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})$ denoted by $\boldsymbol{\alpha}_t \in \mathbb{R}^n$,
- the stochastic estimator of the full gradient of our loss $\boldsymbol{X}^\top \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})$, denoted by $\mathrm{r}_t \in \mathbb{R}^d$.

Algorithm. At the beginning of iteration t, we have access to $\boldsymbol{\alpha}_{t-1}$, r_{t-1} and to the iterate $\boldsymbol{w}_{t-1}$.

Thus equipped, we sample an index i uniformly at random over $\{1, \ldots, n\}$. We then compute the gradient of our loss function for that datapoint, on our iterate, yielding $[\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})]_i = \frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$. We update the stochastic gradient estimator $\boldsymbol{\alpha}_t$ by refreshing the contribution of the i-th datapoint and leaving the other coordinates untouched.

Remark 1. *Coordinate j of our estimator $\boldsymbol{\alpha}_t$ contains the latest sampled one-dimensional derivative of $\frac{1}{n} f_j$.*

To get r_t, we do the same, removing the previous contribution of the i-th datapoint, and adding the refreshed contribution. This allows us not to store the full data-matrix in memory.

The rest of the algorithm continues as the deterministic Frank-Wolfe algorithm from the previous subsection: we find the update direction from $\boldsymbol{s}_t = \mathrm{LMO}(\mathrm{r}_t)$, and we update our iterate using a convex combination of the previous iterate $\boldsymbol{w}_{t-1}$ and $\boldsymbol{s}_t$, whereby our new iterate is feasible.

Remark 2. *Our algorithm requires to keep track of the $\boldsymbol{\alpha}_t$ vector and amounts to keeping one scalar per sample in memory. Our method requires the same small memory caveat as other variance reduced algorithms such as SDCA [40], SAG [41] or SAGA [50]. Despite the resemblance of our gradient estimator to the Stochastic Average Gradient [41], the convergence rate analyses are quite different.*

3.3 Analysis

Preliminary tools

Recall that in our setting, our objective function is $\boldsymbol{w} \mapsto f(\boldsymbol{X}\boldsymbol{w})$, where $f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta}_i)$. We suppose that for all i, f_i is L-smooth, which then implies that f satisfies the following non-standard smoothness condition:

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\bar{\boldsymbol{\theta}})\|_p \leq \frac{L}{n}\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|_p \tag{3.6}$$

for every $p \in [1, \infty]$. Note that in this inequality – unlike in the standard definition of L-smoothness with respect to the ℓ_p norm – the same norm appears on both sides of the inequality. This inequality is proven in Appendix A.1. In particular it follows from Eqn. 3.6 that f is (L/n)-smooth with respect to the ℓ_2 norm.

We therefore have the following quadratic upper bound on our objective function f, valid for all $\boldsymbol{w}, \boldsymbol{v}$ in the domain:

$$\begin{aligned} f(\boldsymbol{X}\boldsymbol{w}) \leq f(\boldsymbol{X}\boldsymbol{v}) + \langle \nabla f(\boldsymbol{X}\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v} \rangle \\ + \frac{L}{2n}\|\boldsymbol{X}\left(\boldsymbol{w} - \boldsymbol{v}\right)\|_2^2 \, . \end{aligned} \tag{3.7}$$

For $p \in \{1, 2, \infty\}$, we define the diameters

$$D_p = \max_{\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}} \|\boldsymbol{X}(\boldsymbol{u} - \boldsymbol{v})\|_p. \tag{3.8}$$

Remark 3. *For $p \in \{1, 2\}$, we have that $D_p^p \leq n D_\infty^p$.*

Worst-Case Convergence Rates for Smooth and Convex Objectives

We state our main result in the L-smooth, convex setting. In this section, we suppose that the f_is are L-smooth and convex and that for all $\boldsymbol{\theta}$, $f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta}_i)$. The objective function f then satisfies Eqn. 3.6 as noted previously.

Theorem 5. *Let $H_0 \overset{def}{=} \|\boldsymbol{\alpha}_0 - \nabla f(\boldsymbol{X}\boldsymbol{w}_0)\|_1$ be the initial error of our gradient estimator and $\boldsymbol{w}_\star \in \mathcal{C}$ a solution to OPT. We run Algorithm 3 with step sizes $\gamma_t = 2/(t+2)$. At time-step $t \geq 2$, the expected primal suboptimality $\mathbb{E}\varepsilon_t = \mathbb{E}[f(\boldsymbol{X}\boldsymbol{w}_t) - f(\boldsymbol{X}\boldsymbol{w}_\star)]$ has the following upper bound*

$$\begin{aligned}\mathbb{E}\varepsilon_t \leq & 2L\left(\frac{D_2^2 + 4(n-1)D_1 D_\infty}{n}\right)\frac{t}{(t+1)(t+2)} \\ & + \frac{2\varepsilon_0 + (2D_\infty H_0 + 64LD_1D_\infty)n^2}{(t+1)(t+2)}\end{aligned} \tag{3.9}$$

Remark 4. *The rate of the proposed method in terms of gradient calls is also given by Eqn. 3.9 (one gradient call per iteration), whereas for deterministic Frank-Wolfe, the (deterministic) suboptimality after t gradient calls has the following upper bound [15, 45]*

$$\varepsilon_t \leq \frac{2LD_2^2}{t}\,. \tag{3.10}$$

In this paper, we will only discuss unit batch size. We can adapt our algorithm and proofs to consider sampling a mini-batch of b datapoints at each step. The leading term in our rate from Theorem 5 will change: we will use $\rho = 1 - \frac{b}{n}$ in Lemma 3. The overall rate will be modified accordingly. The per-iteration complexity will then become $\mathcal{O}(bd)$.

We first **sketch the outline of the proof** before delving into details. The proof of this convergence rate builds on three key lemmas. The first is an adaptation of Lemma 2 of Mokhtari, Hassani, and Karbasi [38] which bounds the suboptimality at step t by the sum of a contraction in the suboptimality at $t-1$, a vanishing term due to smoothness, and a last term depending on our gradient estimator's error in ℓ_1 norm. The first two terms show up in the convergence proof of the full-gradient Frank-Wolfe, see Lacoste-Julien and Jaggi [18]. The last term is an error, or noise term. Supposing the error term vanishes fast enough, we can fall back on the full-gradient proof technique [12, 15].

From there, we show that the error term verifies a particular recursive inequality in lemma 2. In lemma 3, we then leverage this inequality to prove that the error term vanishes as $\mathcal{O}(1/t)$, finally allowing us to obtain the promised rate. The formal statements of these lemmas follow.

Lemma 1. *Let f_i be convex and L-smooth for all i. For any direction $\boldsymbol{\alpha}_t \in \mathbb{R}^n$, define $\boldsymbol{s}_t = \mathrm{LMO}(\boldsymbol{X}^\top \boldsymbol{\alpha}_t)$, $\boldsymbol{x}_t = (1-\gamma_t)\boldsymbol{x}_{t-1} + \gamma_t \boldsymbol{s}_t$ and $H_t = \|\boldsymbol{\alpha}_t - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1$.*

We have the following upper bound on the primal suboptimality at step t:

$$\varepsilon_t \leq \ (1-\gamma_t)\varepsilon_{t-1} + \gamma_t^2 \frac{LD_2^2}{2n} + \underbrace{\gamma_t D_\infty H_t}_{\textit{error term}}\,. \tag{3.11}$$

We defer this proof to Appendix B.3.

Remark 5. *This lemma holds for any direction $\boldsymbol{\alpha}_t \in \mathbb{R}^n$, not necessarily the $\boldsymbol{\alpha}_t$ given by the SFW algorithm.*

Remark 6. *This lemma generalizes the key inequality used in many proofs in the Frank-Wolfe literature [15] but includes an extra error term to account for the fact that the direction $\boldsymbol{\alpha}_t$, which we use for the* LMO *step and therefore to compute the updated iterate, is not the true gradient. If $\boldsymbol{\alpha}_t = \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})$, that is, if we compute the gradient on the full dataset, then $H_t = 0$ and we recover the standard quadratic upper bound.*

In the following, $\boldsymbol{\alpha}_t$ is the direction given by Algorithm 3, and the ℓ_1 error term is in terms of that $\boldsymbol{\alpha}_t$:

$$H_t \stackrel{\text{def}}{=} \|\boldsymbol{\alpha}_t - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1 \tag{3.12}$$

for $t > 0$ and $H_0 = \|\boldsymbol{\alpha}_0 - \nabla f(\boldsymbol{X}\boldsymbol{w}_0)\|_1$.

Notice that we define the gradient estimator's error with the ℓ_1 norm. The previous lemma also holds with the ℓ_2 norm of the gradient error (replacing D_∞ by D_2). We prefer the ℓ_1 norm because of the finite-sum assumption: it induces a coordinate-wise separation over $\boldsymbol{\alpha}_t$ which corresponds to a datapoint-wise separation. The following lemma crucially leverages this assumption to upper bound H_t given by the SFW algorithm.

Lemma 2. *For the stochastic gradient estimator $\boldsymbol{\alpha}_t$ given by Algorithm 3 (SFW), we can upper bound $H_t = \|\boldsymbol{\alpha}_t - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1$ in conditional expectation as follows*

$$\mathbb{E}_t H_t \leq \left(1 - \frac{1}{n}\right)\left(H_{t-1} + \gamma_{t-1}\frac{LD_1}{n}\right). \tag{3.13}$$

Proof. We have the following expression for $\boldsymbol{\alpha}_t$, supposing that index i was sampled at step t.

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \left(\frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}_{t-1}^{(i)}\right)\boldsymbol{e}_i \tag{3.14}$$

where $\boldsymbol{e}_i$ is the i-th vector of the canonical basis of $\mathbb{R}^n$. Consider a fixed coordinate j. Since there is a $\frac{1}{n}$ chance of $\boldsymbol{\alpha}_j$ being updated to $f_j'(\boldsymbol{x}_j^\top \boldsymbol{w}_{t-1})$, taking conditional expectations we have

$$\mathbb{E}_t H_t^j \stackrel{\text{def}}{=} |\boldsymbol{\alpha}_t^{(j)} - \frac{1}{n} f_j'(\boldsymbol{x}_j^\top \boldsymbol{w}_{t-1})| \tag{3.15}$$

$$= \left(1 - \frac{1}{n}\right)|\boldsymbol{\alpha}_{t-1}^{(j)} - \frac{1}{n} f_j'(\boldsymbol{x}_j^\top \boldsymbol{w}_{t-1})|. \tag{3.16}$$

Summing over all coordinates we then have

$$\mathbb{E}_t H_t = \sum_{j=1}^{n} \mathbb{E}_t H_t^j \tag{3.17}$$

$$= \left(1 - \frac{1}{n}\right)\underbrace{\|\boldsymbol{\alpha}_{t-1} - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1}_{\delta_{t-1}}. \tag{3.18}$$

We denote the ℓ_1 norm term by δ_{t-1} for ease. Let us introduce the full gradient at the previous step $\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-2})$ and use the triangle inequality. Our finite sum assumption gives us that for all $j \in \{1, \ldots, n\}$ and $\boldsymbol{w} \in \mathcal{C}$, $[\nabla f(\boldsymbol{X}\boldsymbol{w})]_j = \frac{1}{n} f_j'(\boldsymbol{x}_j^\top \boldsymbol{w})$. Then, we separate the

ℓ_1 norm, use L-smoothness of each of the f_js and the definition of $\boldsymbol{w}_{t-1}$.

$$\delta_{t-1} \leq H_{t-1} + \|\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-2}) - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1 \tag{3.19}$$

$$\leq H_{t-1} + \frac{L}{n}\sum_{j=1}^{n} |\boldsymbol{x}_j^\top(\boldsymbol{w}_{t-1} - \boldsymbol{w}_{t-2})| \tag{3.20}$$

$$\leq H_{t-1} + \gamma_{t-1}\frac{L}{n}\sum_{j=1}^{n} |\boldsymbol{x}_j^\top(\boldsymbol{s}_{t-1} - \boldsymbol{w}_{t-2})| \tag{3.21}$$

$$\leq H_{t-1} + \gamma_{t-1}\frac{L}{n}\|\boldsymbol{X}(\boldsymbol{s}_{t-1} - \boldsymbol{w}_{t-2})\|_1 \tag{3.22}$$

where we used $\boldsymbol{w}_{t-1} - \boldsymbol{w}_{t-2} = \gamma_{t-1}(\boldsymbol{w}_{t-1} - \boldsymbol{s}_{t-2})$. Finally, using the definition of the diameter D_1, we obtain inequality Eqn. 3.13. □

Now, we can use the structure of this recurrence to obtain the desired rate of convergence for our gradient estimator. We state this in the following lemma.

Lemma 3. *Let $\gamma_t = \frac{2}{t+2}$. We have the following bound on the expected error $\mathbb{E}H_t$, for $t \geq 2$:*

$$\mathbb{E}H_t \leq 2\frac{LD_1}{n}\left(\frac{2(n-1)}{t+2} + \left(1 - \frac{1}{n}\right)^{t/2}\log t\right) + \left(1 - \frac{1}{n}\right)^{t} H_0. \tag{3.23}$$

Remark 7. *Our gradient estimator's error in ℓ_1 norm goes to zero as $O\left(\frac{D_1}{t}\right)$. This rate depends on the assumption of the separability of f into a finite sum of L-smooth f_i's. On the other hand, it does not require that each (or any) f_i be convex.*

Proof. Consider a general sequence of nonnegative numbers, $u_0, u_1, u_2, \ldots, u_t \in \mathbb{R}_+$ where for all t, the following recurrence holds:

$$u_t \leq \rho\left(u_{t-1} + \frac{K}{t+1}\right) \tag{3.24}$$

where $0 < \rho < 1$ and $K > 0$ are scalars.

First note that all the iterates are nonnegative. Suppose $t \geq 2$,

$$\begin{aligned} u_t &\leq \rho^t u_0 + K\sum_{k=1}^{t}\frac{\rho^{t-k+1}}{k+1} \\ &= \rho^t u_0 + K\left(\sum_{k=1}^{\lfloor t/2\rfloor}\frac{\rho^{t-k+1}}{k+1} + \sum_{k=\lfloor t/2\rfloor+1}^{t}\frac{\rho^{t-k+1}}{k+1}\right) \\ &\leq \rho^t u_0 + K\left(\sum_{k=1}^{\lfloor t/2\rfloor}\frac{\rho^{t/2}}{k+1} + \sum_{k=\lfloor t/2\rfloor+1}^{t} 2\,\frac{\rho^{t-k+1}}{t+2}\right). \end{aligned}$$

To go from the second line to the third line, we observe that for "old" terms with large steps sizes, we are saved by the higher power in the geometric term. For the more recent terms, the step-size is small enough to ensure convergence. More formally, in the early terms ($1 \leq k \leq \lfloor t/2 \rfloor$), we upper bound ρ^{t-k+1} by $\rho^{t/2}$. In the later terms ($\lfloor t/2 \rfloor + 1 \leq k \leq t$), we upper bound $\frac{1}{k+1}$ by $\frac{2}{t+2}$.

To obtain the full rate, we now study both parts separately. For the first part, we use knowledge of the harmonic series:

$$\rho^{t/2} \sum_{k=1}^{\lfloor t/2 \rfloor} \frac{1}{k+1} \leq \rho^{t/2} \log\left(\frac{t}{2} + 1\right) \tag{3.25}$$

for $t \geq 2$, we can upper bound $\log\left(\frac{t}{2} + 1\right)$ by $\log t$.

For the second part, we use knowledge of the geometric series:

$$\sum_{k=\lfloor t/2 \rfloor + 1}^{t} \rho^{t-k+1} \leq \frac{\rho}{1-\rho}. \tag{3.26}$$

Finally, for $t \geq 2$

$$0 \leq u_t \leq K\left(\frac{\rho}{(1-\rho)}\ \frac{2}{(t+2)} + \rho^{t/2} \log t\right) + \rho^t u_0. \tag{3.27}$$

The expected error $\mathbb{E}H_t$ verifies our general conditions with $u_0 = H_0 = \|\boldsymbol{\alpha}_0 - \nabla f(\boldsymbol{X}\boldsymbol{w}_{-1})\|_1$, defining $\boldsymbol{w}_{-1} \stackrel{\text{def}}{=} \boldsymbol{w}_0$ for the sake of the proof; $\rho = 1 - \frac{1}{n}$ and $K = \frac{2LD_1}{n}$. Specifying these values gives us the claimed bound. □

The remainder of the proof of Theorem 5 follows the usual Frank-Wolfe proofs in the full gradient case, which can be found e.g. in Frank and Wolfe [12] and Jaggi [15]. Here is a brief sketch of these steps: we tie the three key lemmas together, plugging in the bound on $\mathbb{E}H_t$ given by Lemma 3 into the upper bound on the suboptimality at step t given by Lemma 1. By specifying the step size $2/(t+2)$, and scaling the bounds by a factor of $(t+1)(t+2)$, we obtain a telescopic sum, allowing us to upper bound the expected suboptimality at the latest step considered. The details are deferred to Appendix A.3.

Worst-case Convergence Rates for Smooth, Non-Convex Objectives

We start by recalling the definition of the Frank-Wolfe gap:

$$g_t = \max_{\boldsymbol{s} \in \mathcal{C}} \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s}) \rangle. \tag{3.28}$$

Previous work [15] has shown the importance of the Frank-Wolfe gap. In the convex setting, it is a primal-dual gap, and as such, upper bounds both primal and dual suboptimalities. In the general non-convex setting, it is a measure of near-stationarity. We define a stationary

point as any point $\boldsymbol{w}_\star$ such that for all $\boldsymbol{w} \in \mathcal{C}$, $\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_\star), \boldsymbol{X}(\boldsymbol{w} - \boldsymbol{w}_\star)\rangle \geq 0$ [27]. From this definition, it is clear that the Frank-Wolfe gap g_t is zero only at a stationary point.

In this section, we suppose that f_i is L-smooth for i in $\{1, \ldots, n\}$, but not necessarily convex. The following theorem states that we can still obtain a stationary point from Algorithm 3.

Theorem 6. *Let $\boldsymbol{w}_t$ be computed according to Algorithm 3, then*

$$\liminf_{t\to\infty} \mathbb{E}_t g_t = 0, \tag{3.29}$$

where g_t is the Frank-Wolfe gap.

The proof of this result is deferred to Appendix A.6.

3.4 Stopping Criterion

In this section, we define a natural stochastic Frank-Wolfe gap, and explain why it can be used as a stopping criterion.

We recall the definition of the true Frank-Wolfe gap g_t, and define the stochastic Frank-Wolfe gap $\hat{g}_t$ as:

$$g_t = \max_{\boldsymbol{s}\in\mathcal{C}} \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s})\rangle, \tag{3.30}$$

$$\hat{g}_t = \max_{\boldsymbol{s}\in\mathcal{C}} \langle \boldsymbol{\alpha}_t, \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s})\rangle \tag{3.31}$$

for $\boldsymbol{\alpha}_t$ given by SFW.

The Frank-Wolfe gap's properties make estimating it very desirable: when the gap is small for a given iteration of a Frank-Wolfe type algorithm, we can guarantee we are close to optimum (or to a stationary point in the general non-convex case). Unfortunately, in datasets with many samples, and since it depends on the full gradient, computing this gap can be impractical.

The following proposition shows that the stochastic Frank-Wolfe gap estimator resulting from Algorithm 3 can be used as a proxy for the true Frank-Wolfe gap.

Proposition 1. *For $\boldsymbol{\alpha}_t$ given by Algorithm 3, we can bound the distance between the stochastic Frank-Wolfe gap and the true Frank-Wolfe gap as follows:*

$$|g_t - \hat{g}_t| \leq D_\infty H_t, \tag{3.32}$$

which yields the following bound in expectation

$$\begin{aligned}\mathbb{E}|g_t - \hat{g}_t| \leq\ & 2\frac{LD_1D_\infty}{n}\left(\frac{2(n-1)}{t+2} + \left(1 - \frac{1}{n}\right)^{t/2}\log t\right) \\ & + \left(1 - \frac{1}{n}\right)^t D_\infty H_0.\end{aligned} \tag{3.33}$$

We defer the proof to Appendix A.5.

If $\hat{g}_t$ goes to 0, then the true Frank-Wolfe gap will be expected to vanish as well. We therefore propose to use $\hat{g}_t$, which is computed as a byproduct of our SFW algorithm, as a heuristic stopping criterion, but defer a more in-depth theoretical and empirical analysis of this gap to future work.

3.5 Discussion

Table 3.2: Datasets and tasks used in experiments.

Dataset	n	d	κ/n	f_i	$\mathcal{C}$
Breast Cancer	683	10	0.929	$\log(1+\exp(-\boldsymbol{y}_i\boldsymbol{x}_i^\top\boldsymbol{w}))$	$\{\|\boldsymbol{w}\|_1 \leq \lambda,\ \lambda = 5\}$
RCV1	20,242	47,236	0.021	$\log(1+\exp(-\boldsymbol{y}_i\boldsymbol{x}_i^\top\boldsymbol{w}))$	$\{\|\boldsymbol{w}\|_1 \leq \lambda,\ \lambda = 100\}$
California Housing	20,640	8	0.040	$\frac{1}{2}(\boldsymbol{y}_i - \boldsymbol{x}_i^\top\boldsymbol{w})^2$	$\{\|\boldsymbol{w}\|_1 \leq \lambda, \lambda = 0.1\}$

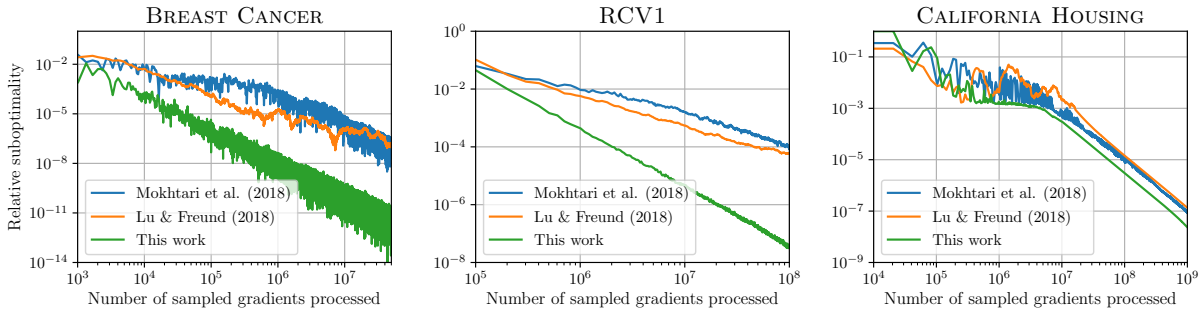


Figure 3.1: Comparing our SFW method to the related works of Lu and Freund [43] and Mokhtari, Hassani, and Karbasi [38]. From left to right: Breast Cancer, RCV1, and California Housing datasets. We plot the relative subtimality values in log-log plots to show empirical rates of convergence. We use the following batch size: $b = \lfloor n/100 \rfloor$.

In this section, we compare the convergence rate of the proposed SFW, Lu and Freund [43] and Mokhtari, Hassani, and Karbasi [38] as shown in Table 3.1. We use big $\mathcal{O}$ notation, only focusing on dependencies in n and t to upper bound the suboptimality at step t.

To make a fair comparison, including dependencies in n, the number of samples, we first standardize notations across papers. Lu and Freund [43] use the same formal setting as ours, where $\boldsymbol{x}_i^\top\boldsymbol{w}$ is the argument to the i-th objective f_i, and the full objective is the average of these. Mokhtari, Hassani, and Karbasi [38] set themselves in a more general setting, where they only assume access to an unbiased estimator of the full gradient.

For ease of comparison, we rewrite the two algorithms of Lu and Freund [43] and Mokhtari, Hassani, and Karbasi [38] in Appendix A.7 using our notations.

Because of their more general setting, the L_{mok} Lipschitz constant appearing in Mokhtari, Hassani, and Karbasi [38] can be written $L_{\text{mok}} = \frac{L}{n} n \max_i \|\boldsymbol{x}_i\|_2$ (using Cauchy-Schwartz). Their diameter constant $D_{\text{mok}} = \max_{\boldsymbol{u},\boldsymbol{v}\in\mathcal{C}} \|\boldsymbol{u}-\boldsymbol{v}\|_2$ is also independent of n. Finally, their σ^2 term controlling the variance of their stochastic estimator should also be n-independent. Under this notation, their convergence rate (Theorem 3, Mokhtari, Hassani, and Karbasi [38]) is $\mathcal{O}\left(1/\sqrt[3]{t}\right)$ with no dependency in n as expected.

Lu and Freund [43] have a detailed discussion of the rate of their method, and achieve the overall rate of $\mathcal{O}(n/t)$.

To fairly compare these rates to the one given by Theorem 5, we must consider the D_1 and D_∞ terms, which may depend on the number of samples n. The rate we obtain has a leading term of $\mathcal{O}(D_1 D_\infty/t)$, and a second term of $\mathcal{O}(D_1 D_\infty n^2/t^2)$. The second term is dominated by the first in the regime $t > n^2$. Defining $\kappa = D_1/D_\infty$, we can write $D_1 D_\infty$ as κD_∞^2. We have that $\kappa \leq n$, meaning that in the worst case, this bound matches the one in Lu and Freund [43]. When the constraint set is the ℓ_1 ball $\{\boldsymbol{w} \,|\, \|\boldsymbol{w}\|_1 \leq \lambda\}$, we have the following closed form expression:

$$\kappa = \frac{\|X\|_{1,1}}{\|X\|_{1,\infty}} = \frac{\max_j \sum_{i=1}^n |X_{ij}|}{\max_{ij} |X_{ij}|}. \tag{3.34}$$

We can therefore easily compute it for given datasets.

Remark 8. *We briefly remark that if for every feature, the contribution of that feature is limited to a few datapoints, this ratio will be small, and therefore the overall bound does not depend on the number of samples. This tends to happen for TF-IDF text representations, and for fat-tailed data.*

Formal analysis of this ratio exceeds the scope of this paper, and we defer it to future work. We report values of κ for the considered datasets in Section 3.7.

3.6 Implementation Details

Our implementation is available in the C-OPT package.[1]

Initialization. We use the cheapest possible initialization: our initial stochastic gradient estimator $\boldsymbol{\alpha}_0$ starts out at 0. We also then have that $\mathrm{r}_0 = 0$.

Sparsity in $\boldsymbol{X}$. Suppose there are at most s non-zero features for any datapoint $\boldsymbol{x}_i$. Then for instances where $\mathcal{C}$ is an ℓ_1 ball, all updates in SFW algorithm can be implemented using using only the support of the current datapoint, making the per-iteration cost of SFW $\mathcal{O}(s)$ instead of $\mathcal{O}(d)$. Large-scale datasets are often extremely sparse, so leveraging this sparsity is crucial. For example, in the LibSVM datasets suite, 8 out of the 11 datasets with more than a million samples have a density between 10^{-4} and 10^{-6}.

[1] `https://github.com/openopt/copt`

3.7 Experiments

We compare the proposed SFW algorithm with other constant batch size algorithms from Mokhtari, Hassani, and Karbasi [38] and Lu and Freund [43].

Experimental Setting. We consider ℓ_1 constrained logistic regression problems on the Breast Cancer and RCV1 datasets, and an ℓ_1 constrained least squares regression problem on the California Housing dataset, all from the UCI dataset repository [51]. See Table 3.2 for details and links.

We compare the relative suboptimality computed for each method, given by $(f(\boldsymbol{X}\boldsymbol{w}_t) - f_{\min})/(f_{\max} - f_{\min})$ at step t, where $f_{\min}$ and $f_{\max}$ are the smallest and largest function values encountered by any of the compared methods. We compute these values at different time intervals (the same for each method) depending on problem size, to limit the time of each run. We use batches using 1% of the dataset at each step, following Lu and Freund [43]. Within a batch, data points are sampled without replacement.

We plot these values as a function of the number of gradient evaluations, equal to the number of iterations times the batch size b: for all of the considered methods, an iteration involves exactly b gradient evaluations and one call to the LMO. This allows us to fairly compare the convergence speeds in practice.

Compared to both methods from Mokhtari, Hassani, and Karbasi [38] and Lu and Freund [43], the proposed SFW achieves lower suboptimality for a given number of iterations on the considered tasks and datasets. We have no explanation for the initial regime in the California Housing dataset, before the methods start showing what resembles a sublinear rate, as the theory prescribes. Notice that the RCV1 dataset has the lowest κ/n (due to sparsity of the TF-IDF represented data), and that the method presented in this paper performs particularly well on this dataset.

Comparison with Mokhtari, Hassani, and Karbasi [38]. Although the step-size in our SFW Algorithm and the one proposed in the paper are of the same order of magnitude $\mathcal{O}(1/t)$, Mokhtari, Hassani, and Karbasi [38] use $f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$ instead of our $(1/n) f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$, because they require an unbiased estimator. Their choice induces higher variance, which then requires the algorithm to use momentum with a vanishing step size in their stochastic gradient estimator, damping the contributions of the later gradients (using $\rho_t = \frac{1}{t^{2/3}}$, see the pseudo code in Appendix A.7). This may explain why the method proposed in Mokhtari, Hassani, and Karbasi [38] achieves slower convergence. On the contrary, the lower variance in our estimator $\boldsymbol{\alpha}_t$ allows us to give the same weight to contributions of later gradients as to previous ones, and to forget all but the last gradient computed at a given datapoint.

Comparison with Lu and Freund [43]. The method from Lu and Freund [43] computes the gradient at an averaged iterate, putting more weight on earlier iterates, making it more conservative. This may explain slower convergence versus the SFW algorithm proposed in this paper in certain settings.

3.8 Conclusion and Future Work

Similarly to methods from the Variance Reduction literature such as SAG, SAGA, SDCA, we propose a Stochastic Frank Wolfe algorithm tailored to the finite-sum setting. Our method achieves a step towards attaining comparable complexity iteration-wise to deterministic, true-gradient Frank-Wolfe in the smooth, convex setting, at a per-iteration cost which can be nearly independent of the number of samples in the dataset in favorable settings. Our rate of convergence depends on the norm ratio κ on the dataset, which is related to a measure of the weights of the data distribution's tails. We will explore this intriguing fact in future work.

We propose a stochastic Frank-Wolfe gap estimator, which may be used as a heuristic stopping criterion, including in the non-convex setting. Its distance to the true gap may be difficult to evaluate numerically. Obtaining a practical bound on this distance is an interesting avenue for future work.

Guélat and Marcotte [17] and Lacoste-Julien and Jaggi [18] have proposed variants of the FW algorithm that converge linearly on polytope constraint sets for strongly convex objectives: the Away Steps Frank-Wolfe and the Pairwise Frank-Wolfe. Goldfarb, Iyengar, and Zhou [44] studied stochastic versions of these and showed linear convergence over polytopes using increasing batch sizes. Our SFW algorithm, the natural stochastic gap and the analyses in this paper should be amenable to such variants as well, which we plan to explore in future work.

Acknowledgments

The authors would like to thank Donald Goldfarb for early encouragement in this direction of research, and Armin Askari, Sara Fridovich-Keil, Yana Hasson, Thomas Kerdreux, Nicolas Le Roux, Romain Lopez, Grégoire Mialon, Courtney Paquette, Hector Roux de Bézieux, Alice Schoenauer-Sebag, Dhruv Sharma, Yi Sun, and Nilesh Tripuraneni for their constructive criticism on drafts of this paper. The authors also warmly thank Maria-Luiza Vladareanu for finding and reporting an error in an earlier draft's proof, and Alex Belloni, Jose Moran for discussions as well.

Francesco Locatello is supported by the Max Planck ETH Center for Learning Systems, by an ETH core grant (to Gunnar Rätsch), and by a Google Ph.D. Fellowship. Robert Freund's research is supported by AFOSR Grant No. FA9550-19-1-0240.

Chapter 4

Constrained Optimization Software

4.1 Introduction

The Python programming language has established itself as the most popular language for scientific computing. Thanks to its high-level interactive nature and its maturing ecosystem of scientific libraries, it is an appealing choice for algorithmic development and exploratory data analysis.

While some of the most-used optimization methods are available in the SciPy [52] package, the Python ecosystem still lacks a comprehensive optimization library. Furthermore, the methods implemented in SciPy can only efficiently handle simple constraints like box-constraints. Finally, these packages are not directly compatible with auto-differentiation libraries such as PyTorch [53] or Jax [54].

We develop two libraries: (COPT) (based on NumPy) and (CHOP (based on PyTorch) to answer this demand.

4.2 Project Vision

The goal of these packages is to develop a comprehensive set of methods for constrained and composite optimization in Python.

Target audience. Scientist and engineers faced with optimization problems that arise in, but are not restricted to, machine learning, signal processing and control.

Code quality. We enforce for code quality by i) maintaining a high code coverage (currently 93% for COPT, 72% for CHOP), using automated code quality assessment tools like pylint (current score=8.47) and having all code reviewed before it's accepted.

Bare-bones design and compatible API. We aim for a similar goal to that of the `scipy.optimize` package: easy usage out of the box. To achieve this, `copt` relies on minimal

dependencies. Our API is as similar to `scipy.optimize`'s API as possible, to offer a smooth learning curve to users. For CHOP, we additionally offer objects with the same API similar as `torch.optim` for stochastic optimizers.

Documentation. Based on sphinx and sphinx-gallery [55]. We strive to provide examples for each method which can be easily extended and adapted by the package's users.

Permissive license. This package is licensed using the New BSD license to encourage its use in both academic and commercial settings. The same licence is used by other related packages like SciPy and scikit-learn, ensuring maximum code reuse.

4.3 Methods Currently Implemented

As of today (August 20, 2023), we have implemented methods in the following families:

- **Proximal methods** These methods optimize composite objectives of the form

$$\min_x f(x) + g(x) \ (+h(x)). \tag{4.1}$$

  Traditional assumptions are that 1) f is smooth, and 2) that we have access to the proximal operator of the typically non-smooth function g (and h). See [56] for an overview.

- **Frank-Wolfe methods** These methods solve optimization problems over convex constraint sets of the form

$$\min_{x \in \mathcal{C}} f(x). \tag{4.2}$$

  Traditional assumptions are that f is smooth and that we have access to a Linear Minimization Oracle (LMO) over $\mathcal{C}$: $\text{LMO}(u) = \arg\min_{v \in \mathcal{C}} \langle u, v \rangle$ [12, 15].

- **Stochastic methods** These methods solve problems using stochastic gradient estimators. These problems are of the form

$$\mathbb{E}_\xi f(x, \xi). \tag{4.3}$$

4.4 Underlying Technologies

- **COPT** We rely on sparse matrix implementations in SciPy [52]. Stochastic methods that need fast iterations use Numba [57] compilation.

- **CHOP** We rely on PyTorch tensor and linear algebra implementations.

4.5 Examples

We provide examples for machine learning model training, signal processing, adversarial examples [39, 58].

See the project websites `https://openopt.github.io/copt/auto_examples/index.html` and `https://github.com/openopt/chop`.

Part II

Models

Chapter 5

Learning differentiable solvers for systems with hard constraints

5.1 Introduction

Methods based on neural networks (NNs) have shown promise in recent years for physics-based problems [59, 60, 61, 62]. Consider a parameterized partial differential equation (PDE), $\mathcal{F}_\phi(u) = \mathbf{0}$. $\mathcal{F}_\phi$ is a differential operator, and the PDE parameters ϕ and solution u are functions over a domain $\mathcal{X}$. Let Φ be a distribution of PDE-parameter functions ϕ. The goal is to solve the following feasibility problem by training a NN with parameters $\theta \in \mathbb{R}^p$, i.e., find θ such that, for all functions ϕ sampled from Φ, the NN solves the feasibility problem,

$$\mathcal{F}_\phi(u_\theta(\phi)) = \mathbf{0}. \tag{5.1}$$

Training such a model requires solving highly nonlinear feasibility problems in the NN parameter space, even when $\mathcal{F}_\phi$ describes a linear PDE.

Current NN methods use two main training approaches to solve Equation 5.1. The first approach is strictly supervised learning, and the NN is trained on PDE solution data using a regression loss [61, 60]. In this case, the feasibility problem only appears through the data; it does not appear explicitly in the training algorithm. The second approach [59] aims to solve the feasibility problem in Equation 5.1 by considering the relaxation,

$$\min_\theta \mathbb{E}_{\phi\sim\Phi} \|\mathcal{F}_\phi(u_\theta(\phi))\|_2^2. \tag{5.2}$$

This second approach does not require access to any PDE solution data. These two approaches have also been combined by having both a data fitting loss and the PDE residual loss [62].

However, both of these approaches come with major challenges. The first approach requires potentially large amounts of PDE solution data, which may need to be generated through expensive numerical simulations or experimental procedures. It can also be challenging to generalize outside the training data, as there is no guarantee that the NN model has learned the relevant physics. For the second approach, recent work has highlighted that in the context of scientific modeling, the relaxed feasibility problem in Equation 5.2 is a difficult optimization

problem [63, 64, 65]. There are several reasons for this, including gradient imbalances in the loss terms [64] and ill-conditioning [63], as well as only *approximate* enforcement of physical laws. In numerous scientific domains including fluid mechanics, physics, and materials science, systems are described by well-known physical laws, and breaking them can often lead to nonphysical solutions. Indeed, if a physical law is only approximately constrained (in this case, "soft-constrained," as with popular penalty-based optimization methods), then the system solution may behave qualitatively differently or even fail to reach an answer.

In this work, we develop a method to overcome these challenges by solving the PDE-constrained problem in Equation 5.1 directly. We only consider the data-starved regime, i.e., we do not assume that any solution data is available on the interior of the domain (however, note that when solution data is available, we can easily add a data fitting loss to improve training). To solve Equation 5.1, we design a PDE-constrained layer for NNs that maps PDE parameters to their solutions, such that the PDE constraints are enforced as "hard constraints." Once our model is trained, we can take new PDE parameters and solve for their corresponding solutions, while still enforcing the correct constraint.

In more detail, our main contributions are the following:

- We propose a method to enforce hard PDE constraints by creating a differentiable layer, which we call *PDE-Constrained-Layer* or PDE-CL. We make the PDE-CL differentiable using implicit differentiation, thereby allowing us to train our model with gradient-based optimization methods. This layer allows us to find the optimal linear combination of functions in a learned basis, given the PDE constraint.

- At inference time, our model only requires finding the optimal linear combination of the fixed basis functions. After using a small number of sampled points to fit this linear combination, we can evaluate the model on a much higher resolution grid.

- We provide empirical validation of our method on three problems representing different types of PDEs. The 2D Darcy Flow problem is an elliptic PDE on a stationary (steady-state) spatial domain, the 1D Burger's problem is a non-linear PDE on a spatiotemporal domain, and the 1D convection problem is a hyperbolic PDE on a spatiotemporal domain. We show that our approach has lower error than the soft constraint approach when predicting solutions for new, unseen test cases, without having access to any solution data during training. Compared to the soft constraint approach, our approach takes fewer iterations to converge to the correct solution, and also requires less training time.

5.2 Background and Related work

The layer we design solves a constrained optimization problem corresponding to a PDE constraint. We outline some relevant lines of work.

Dictionary learning. The problem we study can be seen as PDE-constrained dictionary learning. Dictionary learning [66] aims to learn an over-complete basis that represents the data accurately. Each datapoint is then represented by combining a sparse subset of the learned basis. Since dictionary learning is a discrete method, it is not directly compatible with learning solutions to PDEs, as we need to be able to compute partial derivatives for the underlying learned functions. NNs allow us to do exactly this, as we can learn a parametric over-complete functional basis, which is continuous and differentiable with regard to both its inputs and its parameters.

NNs and structural constraints. Using NNs to solve scientific modeling problems has gained interest in recent years [67]. NN architectures can also be designed such that they are tailored to a specific problem structure, e.g. local correlations in features [68, 4, 69], symmetries in data [70], convexity [71], or monotonicity [72] with regard to input. This reduces the class of models to ones that enforce the desired structure exactly. For scientific problems, NN generalization can be improved by incorporating domain constraints into the ML framework, in order to respect the relevant physics. Common approaches have included adding PDE terms as part of the optimization loss function [59], using NNs to learn differential operators in PDEs such that many PDEs can be solved at inference time [60, 61], and incorporating numerical solvers within the framework of NNs [73]. It is sometimes possible to directly parameterize Gaussian processes [74, 75] or NNs [76] to satisfy PDEs, and fit some desired loss function. However, in the PDEs we study, we cannot have a closed-form parameterization for solutions of the PDE. Previous work in PDE-solving has tried to enforce hard constraints by enforcing boundary conditions [77]. We instead enforce the PDE constraint on the interior domain.

Implicit layers. A deep learning layer is a differentiable, parametric function defined as $f_\theta : x \mapsto y$. For most deep learning layers, the two Jacobians $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial \theta}$ are computed using the chain rule. For these *explicit* layers, f_θ is usually defined as the composition of elementary operations for which the Jacobians are known. On the other hand, *implicit* layers create an implicit relationship between the inputs and outputs by computing the Jacobian using the implicit function theorem [78], rather than the chain rule. Specifically, if the layer has input $x \in \mathbb{R}^{d_{\text{in}}}$, output $y \in \mathbb{R}^{d_{\text{out}}}$ and parameters $\theta \in \mathbb{R}^p$, we suppose that y solves the following nonlinear equation $g(x, y, \theta) = 0$ for some g. Under mild assumptions, this defines an implicit function $f_\theta : x \mapsto y$. In our method, the forward function solves a constrained optimization problem. When computing the Jacobian of the layer, it is highly memory inefficient to differentiate through the optimization algorithm (i.e., all the steps of the iterative solver). Instead, by using the implicit function theorem, a set of linear systems can be solved to obtain the required Jacobians (see Amos and Kolter [5], Barratt [6], Blondel et al. [79], Agrawal et al. [80], and El Ghaoui et al. [81] and the Deep Implicit Layer NeurIPS 2021 tutorial[1] for more details). Implicit layers have been leveraged in many applications, including solving

[1] `http://implicit-layers-tutorial.org/`

ordinary differential equations (ODEs) [82], optimal power flow [83], and rigid many-body physics [84].

Differentiable physics. In a different setting, recent work has aimed to make physics simulators differentiable. The adjoint method [85] is classically used in PDE-constrained optimization, and it has been incorporated into NN training [82, 86, 87]. In this case, the assumption is that discrete samples from a function satisfying an unknown ODE are available. The goal is to learn the system dynamics from data. A NN model is used to approximate the ODE, and traditional numerical integration methods are applied on the output of the NN to get the function evaluation at the next timestep. The adjoint method is used to compute gradients with regard to the NN parameters through the obtained solution. The adjoint method also allows for differentiating through physics simulators [88, 89, 73]. Our setup is different. In our case, the underlying physical law(s) are known and the NN is used to approximate the solutions, under the assumption that there is no observational data in the interior of the solution domain.

5.3 Methods

We describe the details of our method for enforcing PDE constraints within a NN model.

Problem setup

Our goal is to learn a mapping between a PDE parameter function $\phi : \mathcal{X} \to \mathbb{R}$ and the corresponding PDE solution $u(\phi) : \mathcal{X} \to \mathbb{R}$, where the domain $\mathcal{X}$ is an open subset of $\mathbb{R}^d$ for some d. The PDE parameters ϕ could be parameter functions such as initial condition functions, boundary condition functions, forcing functions, and/or physical properties such as wavespeed, diffusivity, and viscosity. We consider well-posed PDEs, following previous work exploring NNs and PDEs [59, 60, 64]. Let $\mathcal{F}_\phi$ be a functional operator such that for all PDE parameter functions ϕ sampled from Φ, the solution $u(\phi)$ satisfies $\mathcal{F}_\phi(u(\phi)) = \mathbf{0}$. The inputs to our NN vary depending on the domain of interest and the PDE parameters. In the simplest case, the input is a pair $(x, \phi(x))$, where $x \in \mathcal{X}$ and $\phi(x)$ is the value of the PDE parameter at x. The output of the NN is the value of the corresponding approximated solution $u_\theta(\phi)$, for a given x. We want to learn the mapping,

$$G : \underbrace{\phi}_{\text{PDE parameters}} \mapsto \underbrace{u(\phi).}_{\text{PDE solutions}} \tag{5.3}$$

We show an example of such a mapping in Figure 5.1. Importantly, we consider only the unsupervised learning setting, where solution data in the interior domain of the PDE is not available for training the model. In this setting, the training is done by only enforcing the PDE, and the initial and/or boundary conditions.

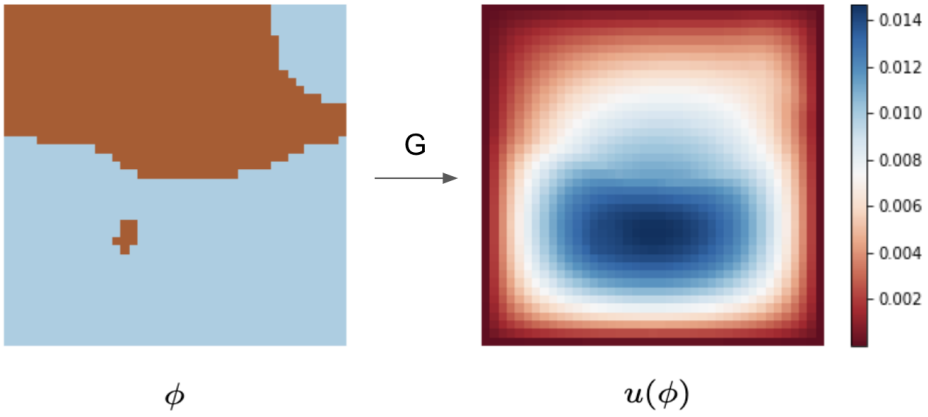


Figure 5.1: **Mapping PDE parameters ϕ to PDE solutions $u(\phi)$.** The goal of our model is to learn a mapping $G : \phi \mapsto u(\phi)$, without access to solution data. As an example, we study the Darcy Flow PDE, which describes the chemical engineering problem of fluid flow through a porous medium [90]. The system is composed of two materials in a given spatial domain $\mathcal{X} = (0, 1)^2$, each with specific diffusion coefficients which depend on the position. The left figure shows ϕ, which encodes the locations and diffusion properties of the two materials. The right figure shows the corresponding solution $u(\phi)$. The function u is a solution of the Darcy Flow PDE with diffusion coefficients ϕ if, for all $(x, y) \in (0, 1)^2$, it satisfies $-\nabla \cdot (\phi(x, y)\nabla u(x, y)) = 1$. The boundary condition is $u(x, y) = 0,\ \forall (x, y) \in \partial(0, 1)^2$.

A differentiable constrained layer for enforcing PDEs

There are two main components to our model. The first component is a NN parameterized by θ, denoted by f_θ. The NN f_θ takes the inputs described in Section 5.3 and outputs a vector in $\mathbb{R}^N$. The output dimension N is the number of functions in our basis, and is a hyperparameter.

The second component of our model, the PDE-constrained layer or PDE-CL, is our main design contribution. We implement a layer that performs a linear combination of the N outputs from the first component, such that the linear combination satisfies the PDE on all points x_i in the discretized domain. Specifically, let ω be the weights in the linear combination given by the PDE-CL. The output of our system is $u_\theta = \sum_{i=1}^{N} \omega_i f_\theta^i$, where f_θ^i is the i-th coordinate output of f_θ. We now describe the forward and backward pass of the PDE-CL.

Forward pass of the differentiable constrained layer. Our layer is a differentiable root-finder for PDEs, and we focus on both linear and non-linear PDEs. As an example, we describe a system based on an affine PDE, where $\mathcal{F}_\phi$ is an affine operator, which depends on ϕ. In our experiments, we study an inhomogeneous linear system in Section 5.4, a homogeneous non-linear system in Section 5.4, and a homogeneous linear system in Section 5.4. The operator $\mathcal{G}_\phi$ is linear when, for any two functions u, v from $\mathcal{X}$ to $\mathbb{R}$, and any scalar $\lambda \in \mathbb{R}$, we

have that,

$$\mathcal{G}_\phi(u + \lambda v) = \mathcal{G}_\phi(u) + \lambda \mathcal{G}_\phi(v). \tag{5.4}$$

The operator $\mathcal{F}_\phi$ is affine when there exists a function b such that the shifted operator $\mathcal{F}_\phi - b$ is linear. Let $\mathcal{G}_\phi$ be the linear part of the operator: $\mathcal{G}_\phi = \mathcal{F}_\phi - b$. We define the PDE-CL to find the optimal linear weighting ω of the N 1D functions encoded by the first NN component, over the set of sampled inputs $x_1, \ldots, x_n$. The vector $\omega \in \mathbb{R}^N$ solves the linear equation system,

$$\forall j = 1, \ldots, n, \quad \mathcal{G}_\phi\left(\sum_{i=1}^{N} \omega_i f_\theta^i\right)(x_j) = b(x_j) \iff \sum_{i=1}^{N} \omega_i \mathcal{G}_\phi(f_\theta^i)(x_j) = b(x_j). \tag{5.5}$$

This linear system is a discretization of the PDE $\mathcal{F}_\phi(u_\theta) = 0$; we aim to enforce the PDE at the sampled points $x_1, \ldots, x_n$. The linear system has n constraints and N variables. These are both hyperparameters, that can be chosen to maximize performance. Note that once N is fixed, it cannot be changed. On the other hand, n can be changed at any time. When the PDE is non-linear, the linear system is replaced by a non-linear least-squares system, for which efficient solvers are available [91, 92].

Backward pass of the differentiable constrained layer. To incorporate the PDE-CL into an end-to-end differentiable system that can be trained by first-order optimization methods, we need to compute the gradients of the full model using an autodiff system [54]. To do this, we must compute the Jacobian of the layer.

The PDE-CL solves a linear system of the form $g(\omega, A, b) = A\omega - b = \mathbf{0}$ in the forward pass, where $A \in \mathbb{R}^{n \times N}$, $\omega \in \mathbb{R}^N$, $b \in \mathbb{R}^n$. Differentiating g with respect to A and b using the chain rule gives the following linear systems, which must be satisfied by the Jacobians $\frac{\partial \omega}{\partial A}$ and $\frac{\partial \omega}{\partial b}$,

$$\forall i, k \in 1, \ldots, N, \quad \forall j \in 1, \ldots, n, \quad 0 = \frac{\partial \omega_i}{\partial A_{jk}} = \left(\omega_k + A_j^\top \frac{\partial \omega}{\partial A_{jk}}\right) \mathbb{1}_{i=j}, \tag{5.6}$$

$$\forall i \in 1, \ldots, N, \quad \forall j \in 1, \ldots, n, \quad 0 = \frac{\partial g_i}{\partial b_j} = A_i^\top \frac{\partial \omega}{\partial b_j} - \mathbb{1}_{i=j}, \tag{5.7}$$

where $\mathbb{1}_{i=j}$ is 1 when $i = j$ and 0 otherwise. Given the size and conditioning of our problems, we cannot directly solve the linear system. Thus, we use an indirect solver (such as conjugate gradient [93] or GMRES [94]) for both the forward system $A\omega = b$ and the backward system given by Equation 5.6 and Equation 5.7. We use the JAX autodiff framework [54, 79] to implement the full model. We include an analysis and additional information on the enforcement of the hard constraints in C.4. We also include an ablation study in C.5 to evaluate the quality of functions in our basis.

Loss function. Our goal is to obtain a NN parameterized function which verifies the PDE over the whole domain. The PDE-CL only guarantees that we verify the PDE over the

sampled points. In the case where $N > n$, the residual over the sampled points $x_1, \ldots, x_n$ is zero, up to numerical error of the linear solver used in our layer. It is preferable to not use this residual for training the NN as it may not be meaningful, and an artifact of the chosen linear solver and tolerances. Instead, we sample new points $x'_1, \ldots, x'_{n'}$ and build the corresponding linear system A', b'. Our loss value is $\|A'\omega - b'\|_2^2$, where ω comes from the PDE-CL and depends on A and b, not A', b'. We compute gradients of this loss function using the Jacobian described above. Another possibility is to use $n > N$. In this case, the residual $\|A\omega - b\|_2^2$ will be non-zero, and while the "hard constraints" will not be satisfied during training, we can minimize this residual loss directly. Let $\mathcal{U}(\mathcal{X})$ denote the uniform distribution over our (bounded) domain $\mathcal{X}$. Formally, our goal is to solve the bilevel optimization problem,

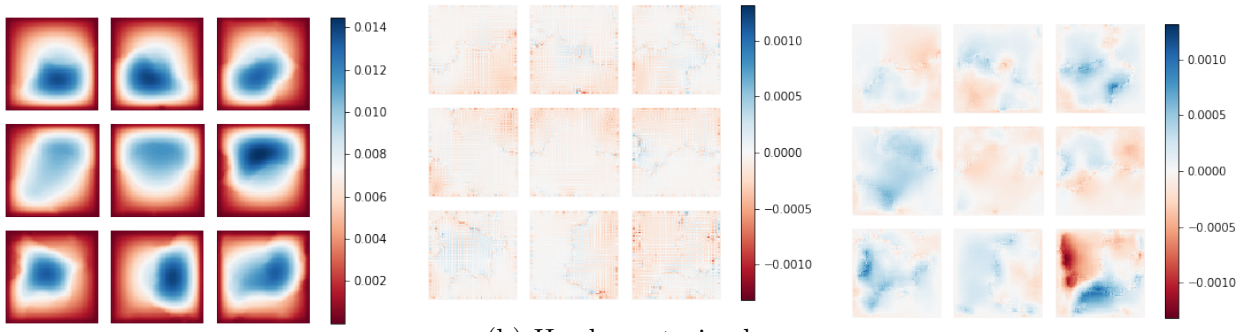
$$\begin{aligned} &\min_{\theta} \ \mathbb{E}_{\phi\sim\Phi}\mathbb{E}_{(x_1,\ldots,x_n),(x'_1,\ldots,x'_n)\sim\mathcal{U}(\mathcal{X})} \ \|A'(\phi, x'_1, \ldots; \theta)\omega(\phi, x_1, \ldots; \theta) - b'(\phi, x'_1, \ldots; \theta)\|_2^2 \\ &\text{s.t. } \omega = \arg\min_{w} \|A(\phi, x_1, \ldots; \theta)w - b(\phi; x_1, \ldots; \theta)\|_2^2. \end{aligned} \tag{5.8}$$

We approximate this problem by replacing the expectations by sums over finite samples. The matrices A, A', and vectors b, b' are built by applying the differential operator $\mathcal{F}_\phi$ to each function in our basis f_θ^i, using the sampled gridpoints. It is straightforward to extend this method in the case of non-linear PDEs by replacing the linear least-squares with the relevant non-linear least squares problem.

Inference procedure. At inference, when given a new PDE parameter test point ϕ, the weights θ are fixed as our function basis is trained. In this paragraph, we discuss guarantees in the linear PDE case. Suppose that we want the values of u_θ over the (new) points $x_1^{\text{test}}, \ldots, x_{n^{\text{test}}}^{\text{test}}$. If $n^{\text{test}} < N$, we can fit ω in the PDE-CL using all of the test points. This guarantees that our model satisfies the PDE on all test points: the linear system in the PDE-CL is underdetermined. In practice, n^{test} is often larger than N. In this case, we can sample $J \subset \{1, \ldots, n^{\text{test}}\}$, $|J| < N$, and fit the PDE-CL over these points. Over the subset $\{x_j^{\text{test}}, j \in J\}$, the PDE will be satisfied. Over the other points, the residual may be non-zero. Another option is to fit the PDE-CL using all the points $x_1^{\text{test}}, \ldots, x_{n^{\text{test}}}^{\text{test}}$, in which case the residual may be non-zero for all points, but is minimized on average over the sampled points by our PDE-CL. **Our method controls the trade-off between speed and accuracy**: choosing a larger N results in larger linear systems, but also larger sets of points on which the PDE is enforced. A smaller N allows for faster linear system solves. Once ω is fit, we can query u_θ for any point in the domain. In practice, we can choose $|J|$ to be much smaller than n^{test}; here, the linear system we need to solve is much smaller than the linear system required by a numerical solver.

5.4 Experimental results and implementation

We test the performance of our model on three different scientific problems: 2D Darcy Flow (Section 5.4), 1D Burgers' equation (Section 5.4), and 1D convection (Section 5.4). In each



(a) Target

(b) Hard-constrained difference

(c) Soft-constrained difference

Figure 5.2: **Heatmaps of Darcy Flow example test set predictions.** We compare our hard-constrained model and the baseline soft-constrained model on a test set of new diffusion coefficients ν. The NN architectures are the same except for our additional PDE-CL in the hard-constrained model. a Target solutions of a subset of PDEs in the test set. b Difference between the predictions of our hard-constrained PDE-CL model and the target solution. c Difference between the predictions of the baseline soft-constrained model and the target solution. Over the test dataset, our model achieves $\mathbf{1.82\% \pm 0.04\%}$ relative error and $\mathbf{0.0457 \pm 0.0021}$ interior domain test loss. In contrast, the soft-constrained model only reaches $\mathbf{3.86\% \pm 0.3\%}$ relative error and $\mathbf{1.1355 \pm 0.0433}$ interior domain test loss. Our model achieves $\mathbf{71\%}$ less relative error than the soft-constrained model. While the heatmaps show a subset of the full test set, the standard deviation across the test set for our model is very low, as shown by the box plot in Appendix C.3.

case, the model is trained without access to any solution data in the interior solution domain. The training set contains 1000 PDE parameters ϕ. The model is then evaluated on a separate test set with $M = 50$ PDE parameters ϕ that are not seen during training. We compare model results on the test set using two metrics: relative L_2 error $\frac{1}{M}\sum_{i=1}^{M}\frac{\|u_\theta(\phi_i)-u(\phi_i)\|_2}{\|u(\phi_i)\|_2}$; and the PDE residual loss $\frac{1}{M}\sum_{i=1}^{M}\|\mathcal{F}_{\phi_i}(u_\theta)\|^2$, which measures how well the PDE is enforced on the interior domain. We demonstrate that our constrained NN architecture generalizes much better on the test set than the comparable unconstrained model for all three problems.[2]

2D Darcy Flow

We look at the steady-state 2D Darcy Flow problem, which describes, for example, fluid flow through porous media. In this section, the PDE parameter (denoted by ϕ in the previous sections as a general variable) is $\nu \in L^\infty((0,1)^2;\mathbb{R}_+)$, a diffusion coefficient. The problem is formulated as follows:

$$\begin{aligned} -\nabla\cdot(\nu(x)\nabla u(x)) &= f(x), \qquad \forall x \in (0,1)^2, \\ u(x) &= 0, \qquad \forall x \in \partial(0,1)^2. \end{aligned} \tag{5.9}$$

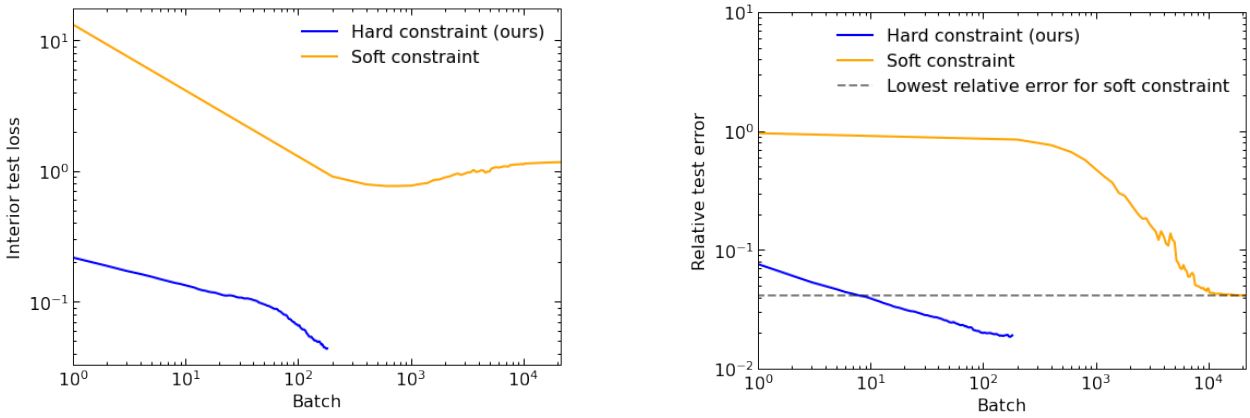
[2]We used a single Titan RTX GPU for each run in our experiments.

Here, f is the forcing function ($f \in L^2((0,1)^2; \mathbb{R})$, and $u(x) = 0$ is the boundary condition. The differential operator is then $F_\nu(u) = -\nabla \cdot (\nu(x) \nabla u(x))$. Given a set of variable coefficients ν, the goal is to predict the correct solution u. The $\nu(x)$ values are generated from a Gaussian and then mapped to two values, corresponding to the two different materials in the system (such as the fluid and the porous medium). We follow the data generation procedure from Li et al. [60]. We use the Fourier Neural Operator (FNO) [60] architecture, trained using a PDE residual loss as the baseline model ("soft-constrained"). Our model uses the FNO architecture and adds our PDE-CL ("hard-constrained"). The domain $(0,1)^2$ is discretized over $n_x \times n_y$ points. For each point on this grid, the model takes as input the coordinates, $x \in (0,1)^2$, and the corresponding values of the diffusion coefficients, $\nu(x)$. The boundary condition is satisfied by using a mollifier function [60], and so the only term in the loss function is the PDE residual. We use a constant forcing function f equal to 1. We provide more details on our setup and implementation in Appendix C.3.

Results. We plot example heatmaps from the test set in Figure 5.2. We compare visually how close our hard-constrained model is to the target solution (Figure 5.2b), and how close the soft-constrained baseline model is to the target solution (Figure 5.2c). Our hard-constrained model is much closer to the target solution, as indicated by the difference plots mostly being white (corresponding to zero difference).

During the training procedure for both hard- and soft-constrained models, we track error on an unseen test set of PDE solutions with different PDE parameters from the training set. We show these error plots in Figure 5.3. In Figure 5.3a, our model starts at a PDE residual test loss value two orders of magnitude smaller than the soft constraint baseline. The PDE residual test loss continues to decrease as training proceeds, remaining significantly lower than the baseline. Similarly, in Figure 5.3b, we show the curves corresponding to the relative error and the PDE residual loss metric on the test dataset. Our model starts at a much smaller relative error immediately and continues to decrease, achieving a final lower test relative error.

On the test set, our model achieves $\mathbf{1.82\% \pm 0.04\%}$ relative error, versus $\mathbf{3.86\% \pm 0.3\%}$ for the soft-constrained baseline model. Our model also achieves $\mathbf{0.0457 \pm 0.0021}$ for the PDE residual test loss, versus $\mathbf{1.1355 \pm 0.0433}$. Our model has almost two orders of magnitude lower PDE residual test loss, and it has significantly lower standard deviation. On the relative error metric, our model achieves a **71% improvement** over the soft-constrained model. While the example heatmaps show a subset of the full test set, the standard deviation across the test set for our model is very low. This indicates that the results are consistent across test samples.



(a) PDE residual loss on test set

(b) Relative error on test set

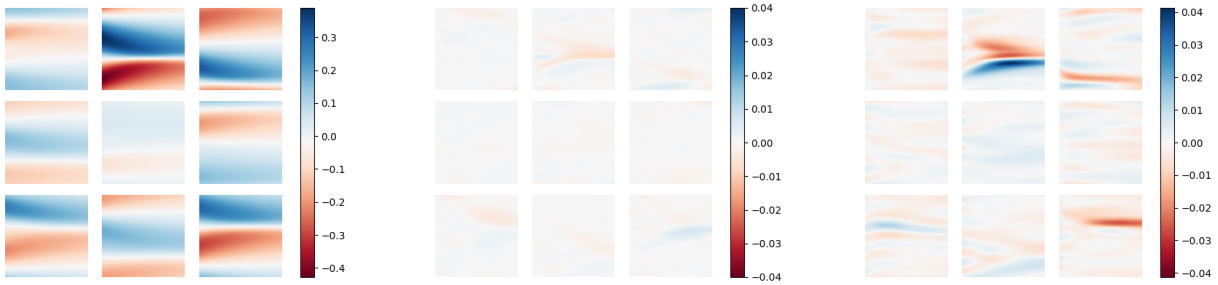
Figure 5.3: **2D Darcy Flow: Error on test set during training.** We train a NN architecture with the PDE residual loss function ("soft constraint" baseline), and the same NN architecture with our PDE-CL ("hard constraint"). During training, we track error on the test set, which we plot on a log-log scale. a PDE residual loss on the test set, during training. This loss measures how well the PDE is enforced. b Relative error on the test set, during training. This metric measures the distance between the predicted solution and the target solution obtained via finite differences. Both measures show that our hard-constrained PDE-CL model starts at a much lower error (over an order of magnitude lower) on the test set at the very start of training, and continues to decrease as training proceeds. This is particularly visible when tracking the PDE residual test loss.

1D Burgers' equation

We study a non-linear 1D PDE, Burgers' equation, which describes transport phenomena. The problem can be written as,

$$\begin{aligned} &\frac{\partial u(x,t)}{\partial t} + \frac{1}{2}\frac{\partial u^2(x,t)}{\partial x} = \nu \frac{\partial^2 u(x,t)}{\partial x^2}, && x \in (0,1), t \in (0,1), \\ &u(x,0) = u_0(x), && x \in (0,1), \\ &u(x,t) = u(x+1,t), && x \in \mathbb{R}, t \in (0,1). \end{aligned} \tag{5.10}$$

Here, u_0 is the initial condition, and the system has periodic boundary conditions. We aim to map the initial condition u_0 to the solution u. We consider problems with a fixed viscosity parameter of $\nu = 0.01$. We follow the data generation procedure from Li et al. [60], which can be found here. We use a physics-informed DeepONet baseline model [64] with regular multi-layer perceptrons as the base NN architecture, trained using the PDE residual loss. Our hard-constrained model is composed of stacked dense layers and our PDE-CL, which allows for a fair comparison. Because this PDE is non-linear, our PDE-CL solves a non-linear least-squares problem, using the PDE residual loss and the initial and boundary condition losses.



(a) Target (b) Hard-constrained difference (c) Soft-constrained difference

Figure 5.4: **Heatmaps of 1D Burgers' example test set predictions.** We compare our hard-constrained model and the baseline soft-constrained model on a test set of new initial conditions u_0. Both architectures are the same, except for our additional PDE-CL in the hard-constrained model. a) Target solutions of a subset of PDEs in the test set. b) Difference between the predictions of our hard-constrained model and the target solution. c) Difference between the predictions of the baseline soft-constrained model and the target solution. Over the test dataset, our model achieves $\mathbf{1.11 \pm 0.11}\%$ relative error. The baseline soft-constrained model achieves only $\mathbf{4.34}\% \pm \mathbf{0.33}\%$ relative error. We use the same base network architecture (MLPs) for both the soft-constrained and hard-constrained model. The errors in both models are concentrated around the "sharp" features in the solution, but these errors have 4x higher magnitude in the soft-constrained model.

Results. We plot example heatmaps from the test set in Figure 5.4. We compare how close our hard-constrained model's output is to the target solution (b)), and similarly for the soft-constrained baseline model (c)). The solution found by our hard-constrained model is much closer to the target solution than the solution found by the baseline model, and our model captures "sharp" features in the solution visibly better than the baseline model. Our hard-constrained model achieves $\mathbf{1.11 \pm 0.11}\%$ relative error after less than $5,000$ steps of training, using just dense layers and our PDE-CL. In contrast, the soft-constrained baseline model only achieves $\mathbf{4.34}\% \pm \mathbf{0.33}\%$ relative error after many more steps of training.

During the training procedure for both hard- and soft-constrained models, we track the relative error on a validation set of PDE solutions with different PDE parameters from the training set. We show the error plot in Figure 5.5. The target solution was obtained using the Chebfun package [95], following Wang, Wang, and Perdikaris [64]. Our model achieves lower error much earlier in training. The large fluctuations are due to the log-log plotting, and small batches used by our method for memory reasons.

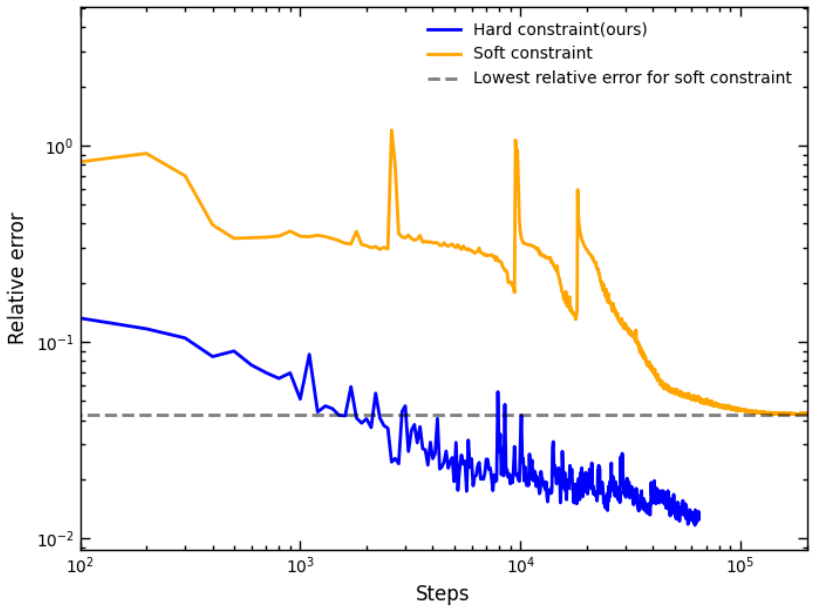


Figure 5.5: **1D Burgers' equation: Error on validation set during training.** We train a NN with the PDE residual loss function ("soft constraint" baseline) and the same NN architecture with our PDE-CL ("hard constraint"). Both architectures are MLPs. During training, we track relative error on the test set, which we plot on a log-log scale. Our hard-constrained model learns low error predictions much earlier in training. The hard constrained model achieves lower relative error than the soft-constrained method.

1D convection

We study a 1D convection problem, describing transport phenomena. The problem can be formulated as follows:

$$\begin{aligned} &\frac{\partial u(x,t)}{\partial t} + \beta(x)\frac{\partial u(x,t)}{\partial x} = 0, && x \in (0,1), t \in (0,1), \\ &h(x) = \sin(\pi x), && x \in (0,1), \\ &g(t) \ = \sin\left(\frac{\pi}{2}t\right), && t \in (0,1). \end{aligned} \tag{5.11}$$

Here, $h(x)$ is the initial condition (at $t = 0$), $g(t)$ is the boundary condition (at $x = 0$), and $\beta(x)$ represents the variable coefficients (denoted by ϕ in Section 5.3). Given a set of variable coefficients, $\beta(x)$, and spatiotemporal points (x_i, t_i), the goal is to predict the correct solution $u(x,t)$. We provide results and more details in Appendix C.1.

5.5 Conclusions

We have considered the problem of mapping PDEs to their corresponding solutions, in particular in the unsupervised setting, where no solution data is available on the interior of the domain during training. For this situation, we have developed a method to enforce hard PDE constraints, when training NNs, by designing a differentiable PDE-constrained layer (PDE-CL). We can add our layer to any NN architecture to enforce PDE constraints accurately, and then train the whole system end-to-end. Our method provides a means to control the trade-off between speed and accuracy through two hyperparameters. We evaluate our proposed method on three problems representing different physical settings: a 2D Darcy Flow problem, which describes fluid flow through a porous medium; a 1D Burger's problem, which describes viscous fluids and a dissipative system; and a 1D convection problem, which describes transport phenomena. Compared to the baseline soft-constrained model, our model can be trained in fewer iterations, achieves lower PDE residual error (measuring how well the PDE is enforced on the interior domain), and achieves lower relative error with respect to target solutions generated by numerical solvers.

Acknowledgements. The authors would like to thank Quentin Berthet, David Duvenaud, Romain Lopez, Dmitriy Morozov, Parth Nobel, Daniel Rothchild, Hector Roux de Bézieux, and Alice Schoenauer Sebag for helpful comments on previous drafts. MWM would like to acknowledge the DOE, NSF, and ONR for providing partial support of this work. This material is based in part upon work supported by the Intelligence Advanced Research Projects Agency (IARPA) and Army Research Office (ARO) under Contract No. W911NF-20-C-0035. ASK and MWM would like to acknowledge the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program under contract No. DE-AC02-05CH11231.

Chapter 6

Probabilistic forecasting with coherent aggregation

6.1 Introduction

Obtaining accurate forecasts is an important step for long-term planning in complex and uncertain environments, with applications ranging from energy management to supply chains, from transportation to climate prediction [96, 97, 98]. Going beyond point forecasts such as means and medians, probabilistic forecasting provides a key tool for forecasting uncertain future events. This involves, e.g., forecasting that there is a 90% chance of rain on a certain day, or that there is a 99% chance that people will want to buy fewer than 100 items at a certain store on a certain week. Providing more detailed predictions of this form permits finer uncertainty quantification. This in turn permits planners to prepare for different scenarios and to allocate resources depending on their anticipated likelihood and cost structure. This can lead to better resource allocation, improved decision making, and less waste.

In many applications, there exist natural hierarchies over quantities one wants to forecast: energy consumption at various temporal granularities, from monthly to weekly; or at different geographic granularities, from the building-level to city-level to state-level; or forecasting retail demand for specific items, as well as for categories of items or brands. Typically, most or all levels of the hierarchy are important: base-levels of the hierarchy are key for operational short-term planning; and higher levels of aggregation yield insights on longer-term or coarser trends. Moreover, it is often desired that the probabilistic forecasts at different granularities are coherent (or consistent) for efficient decision-making at all levels [96, 99].

To be somewhat more precise, we say that probabilistic forecasts at different granularities in a hierarchy are *coherent* if and only if there exists a valid joint distribution across all base-levels such that the probabilistic forecasts have the same distributions as the corresponding marginals of the joint distribution [100, 99, 101]. See also Def. 6.2.1 below. This follows [101], and informally it means that the distribution of an aggregate is the sum of the distributions of the base-levels in the aggregate. Designing a model which is accurate at all levels of aggregation of the hierarchy, and which can exploit information at different levels, while also

enforcing coherency, is well-known to be a difficult challenge [102]. In particular, one can *not* simply aggregate or disaggregate probabilistic forecasts independently (assuming, of course, that one wants to achieve reasonable accuracy), without accounting for correlations among base time series.

In the last few years, end-to-end trainable neural network models have achieved a measure of success for (multi-horizon) probabilistic forecasting for univariate time series [103, 104, 105, 106, 107]. Compared to previous auto-regressive methods, these models provide additional flexibility: one can now fit quantile functions directly through nonlinear quantile regression [104, 103, 108, 106, 109, 107] (while forbidding quantile crossing [110]); and one can differentiate through sampling complex hierarchical graphical models. The added flexibility results in higher forecast accuracy.

In addition to multi-horizon univariate time series forecasting problems, targets to be forecasted sometimes lie in a linear subspace of a common multivariate target. This is the case for hierarchical forecasting: there is a linear relationship between base-level series and aggregates [101, 99]. Despite the flexibility of neural network models, we cannot expect the output of these models to learn from the training data to satisfy (these hierarchical, or other) constraints exactly [111, 10]. Recent work has aimed at enforcing these constraints exactly; and these neural networks models have achieved state of the art results for the *hierarchical* forecasting setup [100, 112, 113]. These end-to-end forecasting models mitigate an important shortcoming of previous (pre-neural network) hierarchical forecasting methods: the need to forecast first, before reconciliating those forecasts in a coherent manner. By exploiting an end-to-end training approach, these methods permit one to train a coherent model in one step: either by integrating the reconciliation as a differentiable module [100]; or by designing a probabilistic model which enforces the coherence, with distribution parameters given by neural networks [112].

In the light of the recent literature, here are properties which a hierarchical forecasting method should satisfy: 1) coherence by construction; 2) end-to-end trainability; 3) ability to optimize for arbitrary sample-based loss functions, e.g., quantile loss, Continuous Ranked Probability Score (CRPS), depending on the use-case; 4) exploitation of the hierarchical structure in the data, leading to a factor model representation; 5) flexibility in the choice of factor and base-level noise distributions that best approximate the data distribution; and 6) compact representation of the forecast to minimize storage cost.

In this paper, we present a method which satisfies all of these properties. We are aware of only two previous methods which provide end-to-end trainable, and coherent hierarchical forecasts: [100] and [112]. However, these two previous methods only satisfy some of the properties stated above. See Table 6.1 for a summary.

In more detail, our **main contributions** are the following.

1. We propose a a novel model for probabilistic forecasting that satisfies all the desired properties stated above. Our model explicitly leverages exchangeability of the base-level targets using a factor model structure. It can be easily adapted to use different factor and base-level distributions, e.g., Gamma, Normal, Truncated Normal, etc. Our model enforces coherent aggregation exactly by construction.

Method	Coherence	Differentiable samples	Arbitrary loss function	Factor model	Arbitrary factor/ base distribution	#Parameters to represent a forecast
Deep HierE2E	✓	✓	✓	✗	✗	Low
DPMN	✓	✗	✗	✓	✗	High
This work	✓	✓	✓	✓	✓	Low

Table 6.1: Desirable properties satisfied by the models Deep HierE2E [100, 112] and DPMN [113] and the proposed method. The ideal method is 1) coherent by construction, 2) differentiable with respect to its parameters for efficient optimization of expected loss functions 3) capable of optimizing arbitrary sample-based loss functions, 4) hierarchical in structure, represented by a factor model, 5) flexible in the choice of factor and base-level distributions, and 6) able to produce compact and expressive forecasts for ease of storing predictions. Our proposed method satisfies all of these desired properties.

2. The factor model parameters are the output of a neural network. We optimize this neural network directly by optimizing for marginal forecast accuracy, using a reparametrization trick. In addition, depending on the use-case, our model can be used for optimizing arbitrary forecast metrics, such as quantile losses, CRPS, mean squared error, or combinations of them.

3. Due to the importance of coherent forecasts in practice, we evaluate our method empirically by comparing against previous coherent end-to-end trainable methods, namely those of [100, 112, 113], on three public datasets. Following previous work, we evaluate on the CRPS metric [114], and we find that our method improves on previous methods by **11.8**-**41.4**% depending on the dataset. We additionally evaluate our mean forecasts using the relative MSE, and find that our method improves on previous methods by **28.9**-**44.1**% on two of three datasets. We analyze CRPS results at different levels of the hierarchy, demonstrating higher or comparable accuracy at all levels on the three datasets. Our model achieves this while providing a more compact forecast representation (an important practical consideration) than previous proposed coherent models.

4. We illustrate the influence of the choice of base-level distributions. Changing distributional assumptions, even in seemingly-minor ways, can have a large impact on accuracy, and the best choice often depends sensitively on the data. Our model has the flexibility to evaluate different modeling choices quickly and easily.

6.2 Background and Related Work

Probabilistic Forecasting

Probabilistic forecasts are usually formulated to quantify future uncertainty to inform decision making. We start by providing an overview of the general probabilistic forecasting problem [97].

We focus on real-valued observations $y \in \mathbb{R}$, with $y \sim Y$, a realization of random variable Y. A forecast distributed as $\hat{Y}$ can be represented by an inverse cumulative density function (CDF) on the real line $\mathbb{R}$. At a given forecast creation date, we assume that we have access to a set of prior information, $\boldsymbol{X}$, which we use to inform our forecasts. This information could be historical observations of the variable we want to predict, static features about the entity, e.g., item characteristics in retail, correlated future features, e.g., future holidays which could influence the energy consumption we're forecasting, etc. We want to output the forecast $\hat{Y}$ which uses the most possible information from $\boldsymbol{X}$ to predict the distribution Y from which y is realized. Without access to the full distribution Y, we typically evaluate quality of $\hat{Y} \mid \boldsymbol{X}$ against a single realization y. If we estimate the conditional mean $\mathbb{E}[Y \mid \boldsymbol{X}]$, we set ourselves in the popular least squares regression setting [115]. If we estimate quantiles of $Y \mid \boldsymbol{X}$, we set ourselves in the quantile regression setting [116, 103, 106].

In practice, our forecasts are represented computationally using an inverse CDF, which depends on distribution parameters. These parameters are themselves the output of an (optimized) parametric function, e.g., a linear or generalized linear model or a neural network model. For example, if we constrain our forecast to be normally distributed, we can use its mean and standard deviation to characterize the full forecasted distribution. This mean and standard deviation will be the output of a parametric model, given covariates as input. More flexible models such as normalizing flows [117] would require more parameters. Armed with a forecast class thus parametrized, we can optimize the fit of the distribution to the observation data. For this, we need a loss function which measures how well a proposed forecast in the forecast class fits the observations.

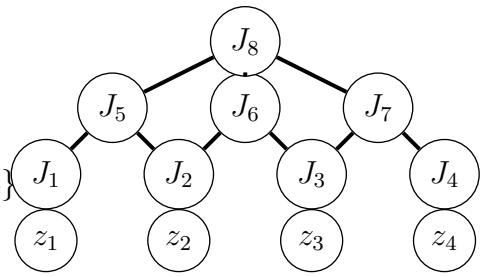
Forecasting with Coherent Hierarchical Aggregation

Hierarchical forecasting aims to solve the probabilistic forecasting problem, as discussed above, but over a set of variables with hierarchical relationships. Most often, we are interested in hierarchies where quantities are summed from one level to the level above it. For instance, the energy consumed in a region is the sum of the energy consumed in each zip code in the region. Methods concerned with hierarchies initially focused on mean forecasts [102, 118, 119, 120, 121, 122, 123, 124]. More recent methods consider probabilistic forecasts [112, 100, 125, 113]. Mean forecasts are easy to aggregate: by linearity of the expectation operator, the mean of the aggregate is the aggregate of the means. However, this is not always the case for other quantities such as quantiles, including the medians.

There are two main method families for hierarchical forecasting: top-down disaggregation [119, 126]; and bottom-up aggregation and reconciliation [102, 118, 127]. Recently, [126]

$J_1 = \{1\}$ $J_5 = \{1, 2\}$
$J_2 = \{2\}$ $J_6 = \{2, 3\}$
$J_3 = \{3\}$ $J_7 = \{3, 4\}$
$J_4 = \{4\}$ $J_8 = \{1, 2, 3, 4\}$

(a) Set representation



(b) Graph representation

$$S = \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \end{array}\right)$$

(c) Matrix representation

Figure 6.1: **A simple hierarchical example of $M = 8$ aggregates over $N = 4$ entities.** Figure 6.1a (left) shows the set representation of the aggregation. Figure 6.1b (center) shows a possible graph representation: the edge between J_6 and J_8 could be removed since J_6 is included in the union of J_5 and J_7. The graph is a DAG, but it is not a tree: the node J_2 appears in the aggregations J_5 and J_6. Figure 6.1c (right) shows the corresponding aggregation matrix. In the matrix representation, we've added horizontal lines to separate *levels* of the hierarchy. These levels do not matter in our algorithm or methods, although they may be important for evaluation. These levels match the levels in the DAG representation, and they correspond to the topological ordering of the nodes in the graph. Our method uses only the matrix representation, which is equivalent to the set representation.

have shown that, in a simplified setting, top-down disaggregation yields provably smaller excess risk than bottom-up aggregation. End-to-end coherent methods [100, 112] (discussed in more detail in Sec. 6.2 below) address the limitation in these two-stage approaches that information is inefficiently used (since models for each series are learned independently and are then post-processed to be coherent). These end-to-end methods fit all levels of the hierarchy simultaneously, with a module for reconciliating forecast at different levels, and they cannot be characterized as either bottom-up or top-down.

End-to-end Coherent Probabilistic Hierarchical Forecasting Models

We are only aware of two methods which yield provably coherent probabilistic forecasts and which allow the models to be trained in an end-to-end manner: Rangapuram et al. [100] and Olivares et al. [112]. Two other works offer guaranteed coherent probabilistic forecasts [126, 127], but they both rely on separately trained models, either for the base-level forecasts or for the top-level forecast. Wang et al. [108] takes a similar approach to ours, motivated by exchangeability between time-series which share global factors, but they do not consider the problem of aggregation. Here, we focus on the two end-to-end trainable coherent forecasting methods [100, 112]. We compare properties of these models with the one

developed in this paper in Table 6.1. The method described in [100] is designed for general convex constraints between multi-variate forecasts. Hierarchical forecasting is a special case of this general problem: it has added structure which is not leveraged by this method. On the other hand, the method described in [112] is too restrictive: it only considers mixtures of Poisson distributions. Since these distributions are integer-valued, this requires training the model using a likelihood loss which cannot directly be related to metrics closer to the downstream goal, such as CRPS or quantile loss on an important quantile level.

Finally, we should note that an important component of our approach to coherent aggregation is to make explicit use of the exchangability of information at the base levels of the hierarchy. To this end, Wang et al. [108]'s observation that exchangeability of base time series induces a factor model structure is related. However, Wang et al. [108] does not consider the hierarchical aggregation setting, and they do not consider end-to-end trainable models, but instead they suppose access to an already trained top-level model. From this perspective, we make broad changes to their method: 1) tailoring it to the coherent aggregation setting; 2) making it end-to-end differentiable; and 3) studying the influence of parametric distributions on accuracy.

End-to-end Probabilistic Hierarchical Forecasting Methods Without Coherence

In a related but separate thread, researchers have designed methods to regularize forecasts to be "more coherent" without enforcing coherence exactly. This is useful for datasets stemming from noisy measurements, e.g., disease control, where local and global measures are taken by different means to estimate quantities at different levels in the hierarchy. In this case, exact hierarchical aggregation does not hold. Han, Dasgupta, and Ghosh [125] focuses on regularizing quantile estimators for coherence between the different levels in the hierarchy. Recently, Kamarthi et al. [113] proposed such a regularization approach for probabilistic forecasts, and focuses on the missing data case. Unlike methods described in the previous paragraph and our method, these methods do not provide guaranteed coherent forecasts between levels of the hierarchy (and thus we do not compare against them).

Notations

We consider the following hierarchical forecasting problem, with a set of base-level entities indexed by $[N] := \{1, \cdots, N\}$. Let $Z_1, \cdots, Z_N \in \mathbb{R}$ be the base-level quantities we want to forecast, and let $\boldsymbol{Z} = (Z_1, \cdots, Z_N)$. In addition, for $n \in [N]$, let $\boldsymbol{X}_n \in \mathbb{R}^D$ be random historical covariates observed for Z_n, and let $\boldsymbol{X} = (\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N) \in \mathbb{R}^{DN}$ be all historical covariates across base time series. At each discrete time step $t \in [T]$, assume that new values of covariates $\boldsymbol{X}$ is given, and new value of targets are observed $\boldsymbol{Z}$. In addition, we introduce another set of indices $[U]$ to extend the framework into a contextual problem. We will refer to this dimension as the item dimension (e.g., in retail) or as the batch dimension. From an applied perspective, there may be multiple attributes or "dimensions" being used to describe the data, and we may or may not be interested in aggregating over all of them. For instance, if we are forecasting

regional item-level demand in retail, then we may want to provide forecasts for many different items in the catalog at multiple regional granularities. The *item* dimension is different than the *region* dimension (over which we want to aggregate in this example). Hence, we reserve the subscript u for the target variables and forecasts corresponding to such a dimension, which we do not use to aggregate across. At each time step t and for each item $u \in [U]$, a forecaster is required to forecast for $\boldsymbol{Z}$, given realized values of $\boldsymbol{X}$. That is, let $z_{t,u,n}$ and $\boldsymbol{x}_{t,u,n}$ be historical observations for the quantity to be forecasted Z_n and covariates $\boldsymbol{X}_n$ for each base-level entity $n \in [N]$. We consider $((z_{t,u,1}, \cdots, z_{t,u,N}), (\boldsymbol{x}_{t,u,1}, \cdots, \boldsymbol{x}_{t,u,N}))$ be a realized value of $(\boldsymbol{Z}, \boldsymbol{X})$.

We can model hierarchical aggregation in terms of a set of base-level entities $[N]$ and various subsets of that set. Suppose that we are interested in M different hierarchical aggregations of the N finest-grained entities. For each $m \in [M]$, we can define the set $J_m \subseteq [N]$ of fine-grained Zs in the m-th aggregate. We suppose that $\{J_1, \cdots, J_M\} \subseteq P([N])$ are subsets of the power set of $[N]$. The target variable corresponding to the m-th aggregate is $Y_m = \sum_{n \in J_m} Z_n$. For instance, if $N = 3$, we could have $J_1 = \{1, 2\}$ and $J_2 = \{2, 3\}$. The set $\{Y_1, \ldots, Y_M\}$ is the set of aggregate targets in which we are interested. It could be that Y_m correspond to a single base-level entity, in which case the corresponding J_m is a singleton.

This setup allows us to define aggregations in matrix form. For $n \in [N]$, let $\boldsymbol{e}_n$ be the n-th canonical basis vector, i.e., the column vector with all zeros except for a 1 in the n-th coordinate. We define the vector $\boldsymbol{s}_m = \sum_{n \in J_m} \boldsymbol{e}_n$. For aggregate m, we have $Y_m = \boldsymbol{s}_m^\top \boldsymbol{Z}$. We therefore define the aggregation matrix

$$S = \left(\boldsymbol{s}_1 \cdots \boldsymbol{s}_M\right)^\top \in \{0, 1\}^{M \times N}. \tag{6.1}$$

If there are U different items, then $y_{t,u,m}$ is the m-th aggregated target variable corresponding to item u at time t. The aggregation matrix S does not depend on u or t in our work. Our method is very general. It applies to the case where we want to aggregate across the region dimension, or the time dimension, or both the region and time dimensions, or other dimensional of the data.

Regardless of the specific hierarchy, we want the probabilistic forecasts to be coherent with respect to the hierarchy. Here is an operational definition.

Definition 6.2.1. Let $\mathbf{Y} = (Y_1, \cdots, Y_M)^\top$ be a multi-variate random variable. Let $\stackrel{d}{=}$ denote equality in distribution. We say that $\mathbf{Y}$ is a *coherent aggregation* of the base-level series $\mathbf{Z}$ for the aggregation matrix S if $\mathbf{Y}$ has the same distribution as $S\mathbf{Z}$, i.e., $\mathbf{Y} \stackrel{d}{=} S\mathbf{Z}$.

Note that historical observations $(y_{t,u,1}, \cdots, y_{t,u,M})$ are always coherent aggregations of base-level observations, as we can view historical observations as degenerate distributions with support at the observed values only.

Our aggregation matrix S defines a directed acyclic graph (DAG), from the base-level entities up. Indeed, any DAG defines a possible hierarchy. In previous work [102, 100, 112, 127], the aggregation graph is assumed to be a tree, since latent variables are associated with each node, either to aggregate up or disaggregate down the forecasted quantities. In our method, it only matters which base-level series are in a given aggregate, i.e., only the aggregation matrix matters. Contrary to previous methods, we do not use relations between

the aggregates (e.g., between Y_m and $Y_{m'}$). Therefore, our method can handle aggregations represented by general DAGs, rather than just trees. In particular, in our method, the aggregates may overlap. See Figure 6.1 for an example, where we consider a hierarchy with $N = 4$ base-level entities and $M = 8$ aggregates. In this hierarchy, we are interested in the base-level entities themselves, in 3 different pairs of the base-level entities, and in the total across all entities. Some of the pairs have a non empty intersection.

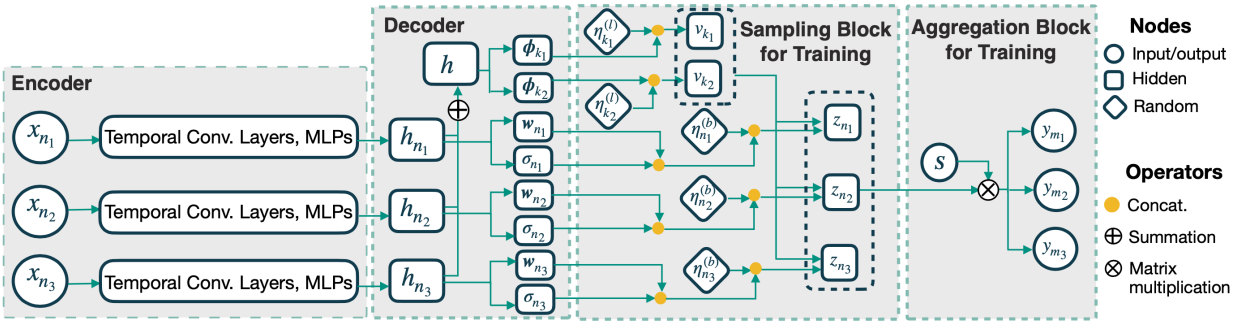


Figure 6.2: Model architecture for our work. We show an example with $N = 3$ base series, $M = 3$ aggregates, and $K = 2$ factors. Base-level series $\boldsymbol{x}_{n_1}, \cdots, \boldsymbol{x}_{n_3}$ are fed into a multi-variate neural network forecasting model such as MQCNN [103]. This model outputs encodings for each base-level series. These encodings are used in two manners: they are summed to produce an encoding at the common factor level, which is decoded into the factor distribution parameters $\boldsymbol{\phi}_{k_1}$ and $\boldsymbol{\phi}_{k_2}$ by a shallow network; they are also decoded directly by another shallow network to produce the base-level distribution loadings $\boldsymbol{w}s$ and parameters $\boldsymbol{\sigma}s$. The base-level forecast distributions can be sampled differentiably using a reparametrization trick by using parameter-free random inputs from factor level $\eta^{(l)}$'s and the base level $\eta^{(b)}$'s. Aggregating these samples $z_{n_1}, z_{n_2}, z_{n_3}$ with aggregation matrices $\boldsymbol{S}$ yields coherent samples at all levels of the hierarchy $y_{m_1}, \ldots, y_{m_3}$. Finally, aggregate samples are used to define the desired loss.

6.3 Our Main Method

In this section, we present our main model for probabilistic forecasting with coherent aggregation.

Factor Model from Exchangeability

We develop a two-level probabilistic factor model for hierarchical forecasting that directly estimates the conditional joint probability function of base-level quantities $Z_1, \cdots, Z_N$ conditioning on historical covariates $\boldsymbol{X}$, i.e.,

$$p(Z_1, \cdots, Z_N \mid \boldsymbol{X}).$$

We can then obtain the target random variables in our hierarchical forecasting problems by marginalizing from this common joint distribution. A reasonable assumption in probabilistic forecasting applications is that the base level forecasts are exchangeable. Indeed, this was *implicitly* used in prior work [112]. Recall that exchangeable random variables are those whose joint probability distribution does not change when the positions in a sequence in which finitely many of them appear are altered.

Under the assumption that $Z_1, \cdots, Z_N$ are exchangeable, conditioned on the covariates $\boldsymbol{X}$, then, motivated by de Finetti's theorem [128, 129, 130, 108], we can model these exchangeable random variables as independent variables, conditioned on some multivariate latent variables $\boldsymbol{v}$, such that

$$p(z_1, \cdots, z_N|\boldsymbol{X}) = \int p^{(l)}(\boldsymbol{v}|\boldsymbol{X}) \left[\prod_{n=1}^{N} p^{(b)}(z_n|\boldsymbol{X}, \boldsymbol{v})\right] d\boldsymbol{v}, \tag{6.2}$$

where $p^{(l)}(\cdot)$ and $p^{(b)}(\cdot)$ are, respectively, the probability functions of latent variable $\boldsymbol{v}$ and of the base-level quantity z, conditioned on the value of latent variable. Choosing the factor distributions and the base-level distributions are important design choices in our model. We will evaluate different choices empirically in section 6.4.

From Eqn. 6.2, we see that the target variables we want to predict are often influenced by common factors. For instance, if we are predicting precipitations in a country over different geographical granularities, then factors like topography may be important (mountainous regions will have different characteristics than coastal regions); and if we are predicting demand in a country, then information about the overall national demand may be important.

Since identifying such factors often requires expert knowledge, we aim to learn them. To do so, we consider the following generic factor structure, where there are K independent random factors V_k for $k \in [K]$, such that

$$p(\boldsymbol{v} \mid \boldsymbol{X}) = \prod_{k=1}^{K} p(v_k \mid \boldsymbol{X}). \tag{6.3}$$

Let each V_k follow a $L^{(l)}$-parameter distribution $f^{(l)}$, with parameters $\boldsymbol{\phi}_k \in \mathbb{R}^{L^{(l)}}$, where $f^{(l)}(\boldsymbol{\phi}_k) \in \mathcal{F}^{(l)}$, the family of chosen distributions, e.g., Gamma, or Normal.

We suppose that the parameters of these factor distributions can be predicted using past observations of the target variables (at all levels of the hierarchy). In particular, let these factors each follow a $L^{(l)}$-parameter distribution, and we define $\boldsymbol{\phi}_k := \phi_k(\boldsymbol{X}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents parameters of a neural network model, and $\phi_k : \mathbb{R}^{DN} \to \mathbb{R}^{L^{(l)}}$. The function ϕ_k maps covariates (representing historical, time-independent and known future features, e.g., day of week) to distribution parameters, e.g., the parameters of a continuous distribution, or parameters of a normalizing flow [117]. The dimension of the image of ϕ_k varies depending on the number of unique parameters required for determining the chosen distribution from the pre-specified distribution family.

Let $\boldsymbol{W} \in [0, 1]^{N \times K}$ be a loading matrix learned with data, and let $\boldsymbol{W} = (\boldsymbol{w}_1^\top, \cdots, \boldsymbol{w}_N^\top)$, where $\forall\ n \in [N]$, $\boldsymbol{w}_n := w_n(\boldsymbol{X}_n; \boldsymbol{\theta})$ and $w_n : \mathbb{R}^N \to [0, 1]^N$. Then each entry in $\boldsymbol{W}$ quantifies

how much the target variable in base-level entity depends on the factors. We assume it can also be predicted from coviariates. Conditioned on a realization $\boldsymbol{v} \sim \boldsymbol{V}$, by Eqn. 6.2, the target variables Z_n for $n \in [N]$ are independent, and each follows a $L^{(b)}-$parameter distribution $f^{(b)}$ from a pre-specified distribution class $\mathcal{F}^{(b)}$. In addition, each distribution depends on parameters $\boldsymbol{\lambda}_n \in \mathbb{R}^{L^{n_{(b)}}}$, where given $\boldsymbol{\sigma}_n := \sigma_n(\boldsymbol{X}_n; \boldsymbol{\theta})$ we let $\boldsymbol{\lambda}_n := \lambda_n(\boldsymbol{w}_n^\top \boldsymbol{v}, \boldsymbol{\sigma}_n)$. $\lambda_n : \mathbb{R}^{L^{(b)}} \to \mathbb{R}^{L^{(b)}}$, the mapping to base-level distribution parameters is assumed to be known, given $\boldsymbol{\sigma}_n$ and $\boldsymbol{w}_n^\top \boldsymbol{v}$; and $\sigma_n : \mathbb{R}^D \to \mathbb{R}^{L^{(b)}-1}$ is learned. Overall, we assume that base-level quantities follow the following two-stage factor model:

$$\begin{aligned} &\forall k \in [K],\ v_k \sim f^{(l)}(\boldsymbol{\phi}_k) \\ &\forall n \in [N],\ z_n \sim f^{(b)}\left(\boldsymbol{\lambda}_n \mid \boldsymbol{w}_n^\top \boldsymbol{v}, \boldsymbol{\sigma}_n\right). \end{aligned} \tag{6.4}$$

With Eqn. 6.3 and Eqn. 6.4, the joint distribution of the $(Z_1, \cdots, Z_N)$ target variables in Eqn. 6.2 can be written as

$$p(z_1, \cdots, z_N | \boldsymbol{X}) = \int \left(\prod_{k=1}^{K} p^{(l)}(v_k | \boldsymbol{\phi}_k) \right) \cdot \prod_{n=1}^{N} p^{(b)}\left(z_n | \boldsymbol{w}_n^\top \boldsymbol{v}, \boldsymbol{\sigma}_n\right) d\boldsymbol{v}. \tag{6.5}$$

Given the general model above, we can compute statistics related to the full joint distribution across multivariate targets in the hierarchy, or marginal distributions of aggregates along the hierarchy. Suppose we are interested in the marginal distribution of Y_m with $J_m = \{n_1, n_2\}$, which contains $n_1, n_2 \in [N]$. Then, the probability function of Y_m can be written as

$$p(y_m) = \int_{z_{n_1} + z_{n_2} = y_m} p(z_1, \cdots, z_n) dz_{n_1} dz_{n_2}. \tag{6.6}$$

In practice, when the integral is not tractable, we can sample from the marginal distribution of the forecasted aggregate $\hat{Y}_m$ by sampling from the base-level series in J_m, and aggregating the samples. In the next sub-section, we discuss how to use these samples to optimize the model for different objective functions.

Sampling and Model Structure

Differentiable sampling of the factor model. We design our method so that it can provide differentiable samples: we can differentiate samples with respect to our model's parameters. Recent work [131, 132, 133] has shown that one can sample in a differentiable manner from almost any continuous distribution. Our method exploits these results: if we can compute differentiable samples from the factor distributions and from the base-level distributions, we can compute differentiable samples for our forecasts, at any level of aggregation.

We now describe how to sample differentiably from the marginal distributions, following Kingma and Welling [134] and Figurnov, Mohamed, and Mnih [131]. To do so, we write our samples as a function of 1) model parameters and 2) samples from a distribution which

does not depend on the model's parameters. More formally, we can write the k-th factor realization, $V_k \mid \boldsymbol{\phi}_k$, and n-th base-level target realization, $Z_n \mid \lambda_n$, as

$$v_k = \alpha^{(l)}(\boldsymbol{\phi}_k, \eta^{(l)}) \tag{6.7}$$

$$z_n = \alpha^{(b)}(\boldsymbol{\lambda}_n, \eta^{(b)}), \tag{6.8}$$

where $\eta^{(l)} \sim \tilde{f}^{(l)}$ is parameter-free noise (e.g., sampled uniformly from $[0, 1]$, or from a standard Normal), and $\alpha^{(l)} : \mathbb{R}^{L^{(l)}+1} \to \mathbb{R}$.

To obtain samples of our forecasted distribution, we start by sampling the factor distributions. The parameters of the factor distributions come from a learned neural network. We use these factor samples as parameters for the base-level distributions. We can then compute samples from the base-level distributions, which are independent conditioned on the (shared) factor samples. Finally, we aggregate the base-level samples up to the desired m-th aggregate. The samples over all aggregates are coherent by construction.

Structure of the model. Expanding on the differentiable sampling module of Eqn. 6.7-6.8, we further explain the overall structure of the model. Let R be total number of samples we want to produce during the training process. For time $t \in [T]$, item $u \in [U]$, $r \in [R]$, let the realized parameter-free noise $\boldsymbol{\eta}^{(l)}_{t,u,r} = (\eta^{(l)}_{t,u,r,1}, \cdots, \eta^{(l)}_{t,u,r,K})$, where all $\eta^{(l)}$s are i.i.d. sampled from $\tilde{f}^{(l)}$. Similarly, let $\boldsymbol{\eta}^{(b)}_{t,u,r} = (\eta^{(b)}_{t,u,r,1}, \cdots, \eta^{(b)}_{t,u,r,N})$, where all $\eta^{(b)}$s are i.i.d. sampled from $\tilde{f}^{(b)}$. Given covariates $\boldsymbol{X}_{t,u}$, we generate parameters for defining the predictive distribution by

$$\boldsymbol{\phi}_{t,u,k} = \phi_k(\boldsymbol{X}_{t,u}; \boldsymbol{\theta}), \qquad \forall\ k \in [K] \tag{6.9}$$

$$\boldsymbol{w}_{t,u,n} = w_n(\boldsymbol{X}_{t,u,n}; \boldsymbol{\theta}), \qquad \forall\ n \in [N] \tag{6.10}$$

$$\boldsymbol{\sigma}_{t,u,n} = \sigma_n(\boldsymbol{X}_{t,u,n}; \boldsymbol{\theta}), \qquad \forall\ n \in [N]. \tag{6.11}$$

Given parameters of the distribution, we further obtain a forecasted sample $\hat{y}_{t,u,r,m}$ for the target $y_{t,u,m}$ by

$$v_{t,u,r,k} = \alpha^{(l)}\left(\boldsymbol{\phi}_{t,u,k}, \eta_{t,u,r,k}\right), \qquad \forall k \in [K] \tag{6.12}$$

$$z_{t,u,r,n} = \alpha^{(b)}\left(\lambda_n(\boldsymbol{w}^\top_{t,u,n}\boldsymbol{v}_{t,u,r}, \boldsymbol{\sigma}_{t,u,n}), \boldsymbol{\eta}^{(b)}_{t,u,r}\right), \qquad \forall\ n \in [N] \tag{6.13}$$

$$\hat{y}_{t,u,r,m} = \boldsymbol{s}_m \boldsymbol{z}_{t,u,r}, \qquad \forall\ m \in [M]. \tag{6.14}$$

Recall that $\boldsymbol{\phi}$s are learned distribution parameters for random factors, $\boldsymbol{w}$s are loadings for these factors on each base-level series, and $\boldsymbol{\sigma}$s are additional parameters for defining distribution at the base level. The vs and zs are the factor-level and base-level predictions in samples; and the ys are the aggregate forecasts in samples. The last line, Eqn. 6.14, ensures that the aggregates are coherent *by construction*.

As shown visually in Figure 6.2, Eqn. 6.9-6.14 define the network structure at training time. In practice, we add some structure to the above-defined ϕ_k and w_n functions: our

network computes encodings $h_1, \ldots, h_n$ of each base-level series. These encodings are then used in two places: 1) we sum them to obtain a top-level encoding $h = \sum_{n=1}^{N} h_n$, which is then decoded by a shallow network into the factor distribution parameters $\boldsymbol{\phi}$; and 2) they are decoded by a separate shallow network into the loadings $\boldsymbol{w}$ and parameters $\boldsymbol{\sigma}$. At inference time, since the predictive distribution is defined by outputs from Eqn. 6.9-6.11, the sampling module Eqn. 6.12-6.14 can be discarded.

Differentiable sampling is implemented for many distributions of interest in several open source machine learning frameworks, e.g., PyTorch[1] [53], TensorFlow[2] [135] and JAX[3] [54], making it extremely easy to implement for many different functional forms. We demonstrate this in the PyTorch code snippet in Figure D.2.1 in Appendix D.2. We only need to change a single line of code to change our distribution assumptions at either level.

Optimizing the Model

Having differentiable samples allows us to optimize for *any* loss which is a differentiable function of forecasted samples. This could be losses on the marginal distributions, i.e., the aggregates, such as squared error loss, quantile losses for quantile levels of interest (0.5 if the median is important, 0.9 or 0.99 if the tails are important), or even weighted combinations of these losses. It could also be a function of the joint distribution, e.g., the energy score [136].

It is common in the hierarchical forecasting literature to evaluate forecast performance on the marginal forecasts [101, 100, 112] using the Continuous Ranked Probability Score (CRPS) [114]. We focus on this loss as an example, and we define it now.

Definition 6.3.1. Let y_m be the target realized value of some underlying distribution Y_m. Let $\hat{Y}_m$ represent the forecasted distribution of Y_m, and let $\hat{y}_m$ and $\hat{y}'_m$ be independent samples of the forecast distribution. Then, the CRPS between the forecasted distribution and observed value, using samples from the forecast distribution, is defined as $\mathbb{E}_{\hat{y},\hat{y}' \sim \hat{Y}_m} \ell_{crps}(\hat{y}, \hat{y}', y_m)$, where

$$\ell_{crps}(\hat{y}, \hat{y}', y_m) = |\hat{y} - y_m| - \frac{1}{2}|\hat{y} - \hat{y}'|. \tag{6.15}$$

The CRPS can also be written as the integral over all quantile levels of the corresponding quantile loss [114]. We use the formulation above because it is an expectation: we can easily produce an unbiased estimator of the CRPS via Monte-Carlo sampling.

Let our model parameters $\boldsymbol{\theta}$ belong to a pre-determined parameter set $\boldsymbol{\Theta}$. Then, as we will be evaluating our model using the CRPS, we fit $\boldsymbol{\theta}$ by optimizing the following objective:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \frac{1}{TUR} \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{u=1}^{U} \sum_{r=1}^{R} \ell_{crps}(\hat{y}_{t,u,r,m}, \hat{y}_{t,u,R+r,m}, y_{t,u,m}), \tag{6.16}$$

where ℓ_{crps} is given by Eqn 6.15.

[1] `https://pytorch.org/docs/stable/distributions.html`: see `rsample` methods.

[2] `https://www.tensorflow.org/probability/api_docs/python/tfp/distributions`

[3] See e.g. `https://jax.readthedocs.io/en/latest/_autosummary/jax.random.gamma.html`.

In practice, we use a neural network with the MQCNN architecture [103] parametrized by $\boldsymbol{\theta}$ to output parameters of the distribution from the covariates $\boldsymbol{X}$. This architecture takes historical, static and future features which can be either numerical or categorical.

Discussion

Here, we compare our method with coherent probabilistic forecasting baselines. We consider the two coherent, end-to-end trainable methods in Rangapuram et al. [100] and Olivares et al. [112]. For completeness, we also consider an ARIMA-based model with reconciliation implemented in [138, 139].

From our perspective, the first method of Rangapuram et al. [100] is "too general." It consists of a neural network model [104] which produces probabilistic forecasts for all time-series in the hierarchy. This method is in fact more general than hierarchical forecasting, since it is designed to enforce any convex constraint satisfied by the forecasts; due to the constraining operation in the method, it has to revise the optimized forecasts. It does not leverage specifics of the hierarchical constraints, which are more structured than a general convex constraint. The model predicts parameters of Gaussian distributions for each time-series in the hierarchy, without coupling. Since the forecasts are not guaranteed to be hierarchically coherent, the model then couples samples from these Gaussian distributions by projecting them on the space of coherent probabilistic forecasts. Both the sampling operation [134] and the projection are differentiable, allowing the method to be trained end-to-end. This model allows different modeling choices, although they are not explored in the initial paper, since Gaussians can be replaced by any distribution which can be sampled in a differentiable way, i.e., almost any continuous distribution [132, 131, 133]. In Rangapuram et al. [100], the projection operator ensures coherence, and correlations between base-levels are learned only by optimizing the neural network. In contrast, our proposed method produces forecasts for base-level series only, while relying on common factors to encode correlations. This removes the need to forecast at all levels simultaneously, therefore reducing computational requirements if we are only interested in a subset of the aggregates.

On the other hand, the method described in Olivares et al. [112] is "too restrictive." It can only handle learned mixture of Poisson distributions. It is in fact a special case of our model. If we suppose that there is a single non-parametric factor V_1 in our model that follows a multivariate discrete distribution with the supports and weights outputted by the neural network, and that conditioning on the realized value of the factor, the base-level distributions are Poisson (Eqn. 6.13), then we recover the model of Olivares et al. [112]. Our model represents a substantial generalization along both directions: we consider multiple independent factors and arbitrary distributions.

Observe that these two methods represent two extremes on the spectrum between non-parametric modeling and parametric modeling. In Rangapuram et al. [100], we do not have access directly to the distributions of the marginals (i.e., the distribution of each aggregate time-series) since the model outputs samples from Gaussian distributions, and then it couples them with a projection. Due to this coupling, the forecasts follow an unknown, non-parametric distribution. On the other hand, in the Olivares et al. [112] method, we can easily describe

the marginals, since they are designed to follow a mixture of Poisson distributions. The same holds true for our proposed method.

Another difference between these two approaches is that the model of Rangapuram et al. [100] optimizes the fit over the marginal time series of interest, under the coherence constraint, while in the model of Olivares et al. [112], the training objective is likelihood-based, which in general does not directly optimize the evaluation metric of interest. In addition, their objective is fixed regardless of the set of marginal time series being evaluated. We design our method so that we can optimize marginal metrics of interest.

Our non-neural network-based baseline ARIMA-MinT-Boot consists of three steps. In the first step, we fit an auto ARIMA model [140] to each marginal time series. Then, we make the mean forecasts coherent by studying the covariance matrix of forecasted errors Wickramasuriya, Athanasopoulos, and Hyndman [123], using ordinary least squares. Lastly, to obtain probabilistic coherent forecasts, we apply a a bootstrap-based method [141] on the coherent point forecasts. Although ARIMA-based methods do not show state-of-the-art performance for these datasets [112], we include it for completeness as an example of a reconciliation method. Among approaches to reconcile point forecasts (such as Wickramasuriya, Athanasopoulos, and Hyndman [123] and Taieb and Koo [127], and approaches to extend them to probabilistic forecasts [141, 101], we only report results for ARIMA-MinT-Boot as they achieved the best CRPS results across most of hierarchical datasets studied in [138].

6.4 Empirical Evaluation

In this section, we present our main empirical results. First, we describe the empirical set up. Second, we evaluate the proposed model, by comparing with two previous end-to-end trainable models, proposed in [100] and [112], as well as an ARIMA-based reconciliation model. Finally, we analyze our model's sensitivity to hyperparameters: 1) choice of base-level distribution; and 2) number of factors.

Setting

Datasets. In our analysis, we consider three qualitatively different (public) datasets: `Tourism-Large`, `Favorita`, and `Traffic`. They have different properties which are representative of more realistic non-public data, and forecasting all of them accurately requires substantial modeling flexibility. The `Tourism-Large` dataset represents the number of visitors to different regions in Australia. The goal is to forecast thousands of visitors, i.e., rescaling count data by 1000. The aggregation is done according to a hierarchy over region and purpose of travel, allowing us to test a case where the aggregate levels have overlap. The `Favorita` dataset is a large retail dataset, and it contains both count data (whole items) and real-valued data (items sold by weight) for over 4000 items. The aggregation hierarchy is regional. We use it to test our method on a (relatively) large-scale problem. Finally, the `Traffic` dataset contains sum-aggregates of highway occupancy rates. The initial rates are hourly, but (following [112]) the dataset we consider is daily, i.e., it uses rates already aggregated

to the daily level for each highway bend as base-level series. The hierarchy in this dataset was defined randomly over highway bends. We use the same hierarchy as previous work. This allows us to test whether our model requires aggregations to be in line with correlation structures to achieve high accuracy. For all three datasets, the forecasted quantities are non-negative.

Pre-processing and features. For data preprocessing, we follow previous work [100, 112]. Our models take in both numerical and categorical features for historical, static and future data, as allowed by the MQCNN architecture [103]. We describe these features in detail in Appendix D.1.

Evaluation metrics. Our main evaluation metric is a target-normalized CRPS [114]. We compute the score described in Eqn. 6.15, and normalize it by dividing the result by the sum of all target values. We also evaluate mean forecasts by reporting ratio between mean squared error across forecasts in all levels over mean squared error of the naive forecast (which treats the previous observation in each time series as the point forecasts for all future horizons), which we call RelMSE.

Hyperparameter search. We consider limited sets of hyperparameters when tuning our model. Since all considered data are non-negative, we consider the Clipped Normal, Truncated Normal, Log-Normal and Gamma distributions as candidates for the base distribution. The Clipped Normal is a normal distribution where all the density at negative values are moved to being point mass at zero. The Truncated Normal on the other hand renormalizes the non-negative part of a normal distribution; there is no added weight on zero. We only consider Gamma random variables as factors, although we could choose any other continuous distribution. When reporting final accuracy results of our model on test set, we used the base distribution that performs the best in validation set, which is Clipped Normal in all three datasets. We determine the number of factors by using results from the validation set. For `Tourism-Large` and `Traffic`, we ran experiments for number of factors $K = \{1, 2, 4, 6, 8, 10, 15, 20, 30, 40\}$; for `Favorita`, due to memory constraints, we set number of factors $K = \{1, 2, 4, 6, 8, 10\}$. Likewise for the learning rate, we performed binary-search by hand on $[10^{-4}, 10^{-3}]$, and we chose the best learning rate according to results on the validation set. This light hyperparameter tuning shows that 1) our method is easy to optimize, and 2) we would get even better results by performing an automated hyperparameter search. Moreover, we only train our models on the CRPS, i.e., we do not yet test the effect of a discrepancy between train and test metrics. We leave this analysis for future work.

Optimization We fit each model using (stochastic) AdamW [142] (Adam [143] with weight decay) using the U dimension as our batch dimension. We use 10^{-5} weight-decay in all of our experiments, without tuning this parameter. `Tourism-Large` and `Traffic` both have a degenerate batch dimension, in the sense that $U = 1$, therefore we use full-batch gradients. For `Favorita`, batch size is 8.

Comparison with Previous Methods

We compare the proposed model to the DPMN model from [112], the HierE2E model from [100], and an ARIMA-based reconcilliation method [123, 141]. Following previous work, we report the CRPS (normalized by the sum of target quantities) at all levels of the defined hierarchies; see Table 6.3 for the best model choices in our proposed family. For detailed definitions of all hierarchical levels of these datasets, see Appendix D.1. The ARIMA-MinT-Boot results are generated using [138], with confidence interval computed based on 10 independent runs. Results for HierE2E is generated based on three independent runs using hyperparameters tuned by [138]. All metrics for DPMN are quoted from [112] with identical experimental setting on all datasets.

Our model achieves lower overall CRPS for the test sets of all three datasets, improving on previous methods by 11.8%, 23.4% and 41.4% on `Tourism-Large`, `Favorita` and `Traffic`, respectively, as seen on the *Overall* rows of Table 6.3. For `Tourism-Large` and `Favorita`, our model achieves better accuracy at almost every single level of the defined hierarchy. On `Traffic`, our model achieves remarkably better results at the finer-grained level (level 3 and 4), but slightly worse accuracy at the higher levels of aggregation. It achieves strong accuracy overall due to its ability to model the fine-grained series very accurately. In this dataset, contrary to the others, the aggregation matrix is defined somewhat randomly, by sampling base-level series. The base-level series within an aggregate therefore do not share special structure, unlike stores in a city for the `Favorita` dataset, or regions in a state in the `Tourism-Large` dataset. This lack of correlation structure may explain the slight performance lag experienced by our model at the higher levels of aggregation. Because RelMSE overweighs aggregated level mean forecast accuracy due to the nature of this metric, our method is associated with sub-optimal mean forecast accuracy overall.

Sensitivity of the Model to Design Choices

We analyze the sensitivity of the model to 1) the base-level distribution, and 2) to the number of chosen factors.

Sensitivity to base-level distribution. To evaluate how accuracy of the proposed model changes depending on the choice of base distribution, we train models for four different base distributions on each dataset, while using Gamma distributions to model the factors V_k. We report the CRPS results in Table 6.4. On all three datasets, Clipped Normal performs the best. On `Tourism-Large` and `Traffic`, Clipped Normal is better than other base-level distributions by a small margin. However, its accuracy is significantly better than others on `Favorita`, possibly due to the base-level series in the retail demand dataset is sparse, and a zero-inflated distribution is needed. For more details, see Appendix D.3. We also observed that Gamma base-level distribution performs worse than Truncated Normal in all three datasets. Finally, on two of the three datasets, the Log-Normal distribution performs poorly.

Sensitivity to numbers of factors. To obtain good results on a large scale dataset such as `Favorita`, [112] required the neural network to output parameters for up to 100 support positions and weights to approximate the empirical distribution of random factors times loadings. Our model allows for several factors, where each factor v_k requires only two parameters, contained in $\phi_k(\boldsymbol{X}; \boldsymbol{\theta})$ (Eqn. 6.9, 6.12). Separating random factors and deterministic factor loadings rather than learning an empirical distribution approximation as in [112], our model can model more complex distributions while requiring fewer parameters. For example, we discovered that using one Gamma factor in our work already yields improved forecast accuracy for `Favorita`, compared to previous models. Moreover, we show how the choice of the number of factors impacts forecast accuracy, using an experiment on `Traffic` dataset as an example. We show the results in Figure 6.3 below. On this dataset, we observe that a single factor already performs better than previous methods. The best results are obtained with 20 factors.

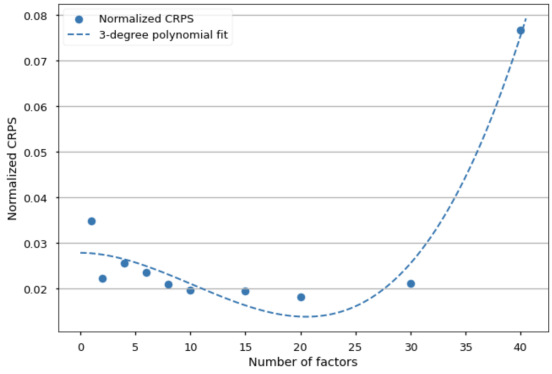


Figure 6.3: Performance of our model for different numbers of factors based on one run with the same random seed. We provide overall normalized CRPS on `Traffic` for a model with Gamma factors and Clipped Normal base distributions. We also fit a 3-degree polynomial function to the results.

6.5 Conclusion

In this work, we propose a novel probabilistic forecasting framework for hierarchical forecasting problems. To guarantee that probabilistic predictions are coherent in aggregation, our framework assumes that the predictive joint distribution over base-level targets follows a factor model, provided that base targets are exchangeable.

While the factor model assumption constrains the predictions, our model leverages recent advances in differentiable sampling, and it can optimize various sample-based objective functions that are aligned with forecasting evaluation metrics. We further conducted experiments on models in this framework on three benchmark datasets, comparing with alternative approaches. We demonstrate that our model improves overall forecast accuracy by 11.8-41.4%

on three benchmark datasets. The model generates best or comparable forecast accuracy on almost all hierarchical levels for three datasets.

When forecasting for a given aggregate quantity within the hierarchy, our proposed model requires data for all base-level series included in the aggregate to be fitted on a single GPU. This may be prohibitive in certain applications where there exists thousands or millions of time series within an hierarchy. Removing this restriction would allow our method to handle to large-scale hierarchies. Our model currently uses a simple aggregation over all base-level embeddings for learning the parameters for the factors, this simple operator can lead to sub-optimal performance when forecasting at the most aggregated levels, as shown in empirical results. Leveraging other neural network structures with higher degrees of freedom for replacing the aggregation operator is a interesting future research direction.

Dataset	# Items (U)	Base (N)	Levels	Aggregated (M)	Time range	Frequency	Horizon (time steps)
`Tourism-Large`	1	304	4/5	555	1998-2016	Monthly	12
`Favorita`	4036	54	4	93	1/2013 - 8/2017	Daily	34
`Traffic`	1	200	4	207	1/2008-3/2009	Daily	1

Table 6.2: Summary of publicly-available data used in our empirical evaluation. The `Tourism-Large` dataset [123] represents tourism visits to Australia between 1998 and 2016. The `Favorita` dataset [137] represents daily grocery sales in stores owned by the Favorita Corporación in Ecuador between 2013 and 2017. The `Traffic` dataset [127] consists of daily occupancy rate for 200 selected car lanes in California Bay Area between 2008 and 2009.

Dataset	Metric	Level	Ours	DPMN-GroupBU	DPMN-NaiveBU	HierE2E	ARIMA-MinT-Boot
`Tourism -Large`	CRPS	Overall	$\mathbf{0.1101 \pm 0.0009}$	0.1249 ± 0.0020	0.1274 ± 0.0028	0.1456 ± 0.0061	0.1317 ± 0.0008
		Level 1	0.0349 ± 0.0028	0.0431 ± 0.0042	0.0514 ± 0.0030	0.0721 ± 0.0095	$\mathbf{0.0277 \pm 0.0011}$
		Level 2	$\mathbf{0.0601 \pm 0.0021}$	0.0637 ± 0.0032	0.0705 ± 0.0026	0.0943 ± 0.0059	0.0628 ± 0.0010
		Level 3	$\mathbf{0.0959 \pm 0.0027}$	0.1084 ± 0.0033	0.1068 ± 0.0019	0.1280 ± 0.0077	0.1150 ± 0.0008
		Level 4	$\mathbf{0.1372 \pm 0.0022}$	0.1554 ± 0.0025	0.1507 ± 0.0014	0.1682 ± 0.0083	0.1688 ± 0.0007
		Level 5	$\mathbf{0.0552 \pm 0.0016}$	0.0700 ± 0.0038	0.0907 ± 0.0061	0.0992 ± 0.0133	0.0750 ± 0.0014
		Level 6	$\mathbf{0.1000 \pm 0.0009}$	0.1070 ± 0.0023	0.1175 ± 0.0047	0.1361 ± 0.0069	0.1221 ± 0.0014
		Level 7	$\mathbf{0.1652 \pm 0.0008}$	0.1887 ± 0.0032	0.1836 ± 0.0038	0.1996 ± 0.0072	0.1976 ± 0.0010
		Level 8	$\mathbf{0.2321 \pm 0.0036}$	0.2629 ± 0.0034	0.2481 ± 0.0026	0.2670 ± 0.0073	0.2841 ± 0.0008
	RelMSE	Overall	$\mathbf{0.0601 \pm 0.0019}$	0.1113 ± 0.0158	0.2680 ± 0.0748	0.2058 ± 0.0443	0.1075
`Favorita`	CRPS	Overall	$\mathbf{0.3080 \pm 0.0201}$	0.4020 ± 0.0182	0.5301 ± 0.0120	0.5099 ± 0.1080	0.3968 ± 0.0007
		Country	$\mathbf{0.2116 \pm 0.0117}$	0.2760 ± 0.0149	0.4166 ± 0.0195	0.3521 ± 0.1526	0.2652 ± 0.0005
		State	$\mathbf{0.2910 \pm 0.0205}$	0.3865 ± 0.0207	0.5128 ± 0.0108	0.4860 ± 0.1120	0.3782 ± 0.0006
		City	$\mathbf{0.3095 \pm 0.0205}$	0.4068 ± 0.0206	0.5317 ± 0.0115	0.5293 ± 0.1046	0.4028 ± 0.0008
		Store	$\mathbf{0.4201 \pm 0.0162}$	0.5387 ± 0.0253	0.6594 ± 0.0150	0.6723 ± 0.0630	0.5410 ± 0.0010
	RelMSE	Overall	$\mathbf{0.5381 \pm 0.0368}$	0.7563 ± 0.0713	0.9533 ± 0.0201	1.082 ± 0.5473	1.1270
`Traffic`	CRPS	Overall	$\mathbf{0.0181 \pm 0.0028}$	0.0907 ± 0.0024	0.0704 ± 0.0014	0.0309 ± 0.0062	0.0731 ± 0.0050
		Level 1	0.0176 ± 0.0030	0.0397 ± 0.0044	$\mathbf{0.0134 \pm 0.0022}$	0.0157 ± 0.0327	0.0452 ± 0.0066
		Level 2	0.0176 ± 0.0030	0.0537 ± 0.0024	0.0289 ± 0.0017	$\mathbf{0.0103 \pm 0.0099}$	0.0470 ± 0.0057
		Level 3	$\mathbf{0.0176 \pm 0.0031}$	0.0538 ± 0.0022	0.0290 ± 0.0011	0.0160 ± 0.0093	0.0544 ± 0.0042
		Level 4	$\mathbf{0.0195 \pm 0.0028}$	0.2155 ± 0.0022	0.2101 ± 0.0008	0.0881 ± 0.0094	0.1459 ± 0.0047
	RelMSE	Overall	0.0232 ± 0.0536	0.1750 ± 0.0099	0.0168 ± 0.0026	$\mathbf{0.0047 \pm 0.0054}$	0.0624

Table 6.3: Results of our empirical evaluation. We report the CRPS score for each dataset (smaller is better) at various hierarchical levels (a level with lower number represents a more aggregated level, more details discussed in Appendix D.1). Average accuracy and its interval are computed based on three independent runs. Our model improves on previous methods on all datasets at all levels but Level 1 of `Traffic`. On `Tourism-Large`, our model improves on the previous state of the art by 11.8%. On the larger-scale `Favorita` dataset, our model improves by 23.4%. On `Traffic`, we improve the best result by 41.4% overall. Our model performs notably better at the finest granularity; our model's performance is stable across levels, whereas the other models perform better at the aggregate levels than at the base level. We evaluate our mean forecasts by computing the RelMSE score at the overall level (summing the total squared error at all levels, and normalize it by that of naive forecasts). Note ARIMA-MinT-Boot produces deterministic mean forecasts across model runs. Our model achieves lower RelMSE than other models on two datasets: 44.1% improvement on `Tourism-Large`, 28.9% on `Favorita`, but reaches higher RelMSE on `Traffic`, despite large gains in CRPS.

	`Tourism-Large`	`Favorita`	`Traffic`
Gamma	0.1174 ± 0.0044	0.4817 ± 0.2274	0.0738 ± 0.0629
Log-Normal	0.2245 ± 0.0905	0.5268 ± 0.1211	1.0135 ± 0.2967
Trunc-Normal	0.1123 ± 0.0038	0.3922 ± 0.0695	0.0216 ± 0.0033
Clipped Normal	$\mathbf{0.1101 \pm 0.0009}$	$\mathbf{0.3080 \pm 0.0201}$	$\mathbf{0.0181 \pm 0.0028}$

Table 6.4: Performance of our model for various choices of base distribution, where results are based on three independent runs. We provide overall normalized CRPS for factor models with Gamma distributed factors and various base distributions.

Bibliography

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc.

Oord, Aaron van den et al. (2016). *WaveNet: A Generative Model for Raw Audio*. cite arxiv:1609.03499.

Bronstein, Michael M. et al. (2017). "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34, pp. 18–42.

Amos, Brandon and J. Zico Kolter (2017a). "OptNet: Differentiable Optimization as a Layer in Neural Networks". In: *ICML*.

Barratt, Shane T. (2018). "On the Differentiability of the Solution to Convex Optimization Problems". In: *arXiv: Optimization and Control*.

Braun, Gábor et al. (Nov. 2022). *Conditional Gradient Methods*. arXiv: 2211.14103 [math.OC]. URL: https://conditional-gradients.org/.

Pedregosa, Fabian et al. (2020). "Linearly Convergent Frank-Wolfe with Backtracking Line-Search". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, pp. 1–10.

Negiar, Geoffrey et al. (2020). "Stochastic Frank-Wolfe for Constrained Finite-Sum Minimization". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 7253–7262. URL: https://proceedings.mlr.press/v119/negiar20a.html.

Négiar, Geoffrey, Michael W. Mahoney, and Aditi Krishnapriyan (2023). "Learning differentiable solvers for systems with hard constraints". In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=vdv6CmGksr0.

Negiar, Geoffrey et al. (2023). "Probabilistic Forecasting with Coherent Aggregation". In: *ArXiv* abs/2307.09797. URL: https://api.semanticscholar.org/CorpusID:259982821.

Frank, Marguerite and Philip Wolfe (1956). "An algorithm for quadratic programming". In: *Naval Research Logistics (NRL)*.

Levitin, Evgeny S and Boris T Polyak (1966). "Constrained minimization methods". In: *USSR Computational mathematics and mathematical physics*.

Demyanov, Vladimir and Aleksandr Rubinov (1967). "The minimization of a smooth convex functional on a convex set". In: *SIAM Journal on Control*.

Jaggi, Martin (2013). "Revisiting Frank-Wolfe: projection-free sparse convex optimization". In: *International Conference on Machine Learning*.

Canon, Michael D and Clifton D Cullum (1968). "A tight upper bound on the rate of convergence of Frank-Wolfe algorithm". In: *SIAM Journal on Control*.

Guélat, Jacques and Patrice Marcotte (1986). "Some comments on Wolfe's 'away step'". In: *Mathematical Programming*.

Lacoste-Julien, Simon and Martin Jaggi (2015). "On the global linear convergence of Frank-Wolfe optimization variants". In: *Advances in Neural Information Processing Systems*.

Beck, Amir, Edouard Pauwels, and Shoham Sabach (2015). "The cyclic block conditional gradient method for convex optimization problems". In: *SIAM Journal on Optimization*.

Dunn, Joseph C (1980). "Convergence rates for conditional gradient sequences generated by implicit step length rules". In: *SIAM Journal on Control and Optimization*.

Locatello, Francesco et al. (2017). "A Unified Optimization View on Generalized Matching Pursuit and Frank-Wolfe". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*.

Garber, Dan and Elad Hazan (2013). "A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization". In: *arXiv preprint arXiv:1301.4666*.

Mitchell, BF, Vladimir Demyanov, and VN Malozemov (1974). "Finding the point of a polyhedron closest to the origin". In: *SIAM Journal on Control*.

Platt, John (1998). "Sequential minimal optimization: A fast algorithm for training support vector machines". In.

Mallat, Stéphane G and Zhifeng Zhang (1993). "Matching pursuits with time-frequency dictionaries". In: *IEEE Transactions on signal processing*.

Locatello, Francesco et al. (2018). "On Matching Pursuit and Coordinate Descent". In: *Proceedings of the 35th International Conference on Machine Learning*.

Bertsekas, Dimitri P (1999). *Nonlinear programming*. Athena Scientific.

Lacoste-Julien, Simon (2016). "Convergence rate of Frank-Wolfe for non-convex objectives". In: *arXiv preprint arXiv:1607.00345*.

Reddi, Sashank J et al. (2016). "Stochastic Frank-Wolfe methods for nonconvex optimization". In: *54th Annual Allerton Conference on Communication, Control, and Computing*.

Nesterov, Yurii (2017). "Complexity bounds for primal-dual methods minimizing the model of objective function". In: *Mathematical Programming*.

Guyon, Isabelle et al. (2008). *Feature extraction: foundations and applications*. Vol. 207. Springer.

Lewis, David D et al. (2004). "RCV1: A new benchmark collection for text categorization research". In: *Journal of machine learning research*.

Harper, F Maxwell and Joseph A Konstan (2015). "The movielens datasets: History and context". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)*.

Mehta, Bhaskar, Thomas Hofmann, and Wolfgang Nejdl (2007). "Robust collaborative filtering". In: *Proceedings of the 2007 ACM conference on Recommender systems*.

Mareček, Jakub, Peter Richtárik, and Martin Takáč (2017). "Matrix completion under interval uncertainty". In: *European Journal of Operational Research*.

Lacoste-Julien, Simon et al. (2013). "Block-Coordinate Frank-Wolfe Optimization for Structural SVMs". In: *Proceedings of the 30th International Conference on Machine Learning*.

Kerdreux, Thomas, Fabian Pedregosa, and Alexandre d'Aspremont (2018). "Frank-Wolfe with Subsampling Oracle". In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR.

Mokhtari, Aryan, Hamed Hassani, and Amin Karbasi (2018). "Stochastic Conditional Gradient Methods: From Convex Minimization to Submodular Maximization". In: *arXiv*. arXiv: `1804.09554v1 [math.OC]`.

Tibshirani, R. (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society (Series B)*.

Shalev-Shwartz, Shai and Tong Zhang (2013). "Stochastic dual coordinate ascent methods for regularized loss minimization". In: *Journal of Machine Learning Research*.

Schmidt, Mark, Nicolas Le Roux, and Francis Bach (2013). "Minimizing finite sums with the stochastic average gradient". In: *Mathematical Programming*.

Hofmann, Thomas et al. (2015). "Variance reduced stochastic gradient descent with neighbors". In: *Advances in Neural Information Processing Systems*.

Lu, Haihao and Robert M Freund (2018). "Generalized Stochastic Frank-Wolfe Algorithm with Stochastic "Substitute" Gradient for Structured Convex Optimization". In: *arXiv*.

Goldfarb, Donald, Garud Iyengar, and Chaoxu Zhou (2017). "Linear Convergence of Stochastic Frank Wolfe Variants". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*.

Hazan, Elad and Haipeng Luo (2016). "Variance-reduced and projection-free stochastic optimization". In: *International Conference on Machine Learning*.

Locatello, Francesco et al. (2019). "Stochastic Frank-Wolfe for Composite Convex Minimization". In: *Advances in Neural Information Processing Systems 32*.

Zhang, Mingrui et al. (2019a). "One Sample Stochastic Frank-Wolfe". In: *arXiv*.

Abernethy, Jacob D. and Jun-Kun Wang (2017). "On Frank-Wolfe and Equilibrium Computation". In: *Advances in Neural Information Processing Systems 30*.

Abernethy, Jacob et al. (2018). "Faster Rates for Convex-Concave Games". In: *Proceedings of the 31st Conference On Learning Theory*.

Defazio, Aaron, Francis Bach, and Simon Lacoste-Julien (2014). "SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives". In: *Advances in Neural Information Processing Systems 27*.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: `http://archive.ics.uci.edu/ml`.

Virtanen, Pauli et al. (2019). "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python". In: *arXiv preprint arXiv:1907.10121*.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Bradbury, James et al. (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. URL: http://github.com/google/jax.

Nájera, Óscar et al. (2020). *sphinx-gallery*. Version v0.7.0. DOI: 10.5281/zenodo.3838216. URL: https://doi.org/10.5281/zenodo.3838216.

Parikh, Neal and Stephen P. Boyd (2014). "Proximal Algorithms". In: *Found. Trends Optim.* 1, pp. 127–239.

Lam, Siu Kwan, Antoine Pitrou, and Stanley Seibert (2015). "Numba: A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy (2014). "Explaining and Harnessing Adversarial Examples". In: *CoRR* abs/1412.6572. URL: https://api.semanticscholar.org/CorpusID:6706414.

Raissi, M., P. Perdikaris, and G.E. Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Li, Zongyi et al. (2020). "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895*.

Lu, Lu et al. (2021a). "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nat. Mach. Intell.* 3, pp. 218–229.

Li, Zongyi et al. (2021). "Physics-informed neural operator for learning partial differential equations". In: *arXiv preprint arXiv:2111.03794*.

Krishnapriyan, Aditi et al. (2021). "Characterizing possible failure modes in physics-informed neural networks". In: *Advances in Neural Information Processing Systems* 34.

Wang, Sifan, Hanwen Wang, and Paris Perdikaris (2021). "Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets". In: *Science Advances* 7.40.

Edwards, Chris (2022). "Neural networks learn to speed up simulations". In: *Communications of the ACM* 65.5, pp. 27–29.

Mairal, Julien et al. (2009). "Online Dictionary Learning for Sparse Coding". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, pp. 689–696. ISBN: 9781605585161. DOI: 10.1145/1553374.1553463. URL: https://doi.org/10.1145/1553374.1553463.

Willard, Jared et al. (2020). "Integrating physics-based modeling with machine learning: A survey". In: *arXiv preprint arXiv:2003.04919* 1.1, pp. 1–34.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86, pp. 2278–2324.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9, pp. 1735–1780.

Cohen, Taco and Max Welling (2016). "Group Equivariant Convolutional Networks". In: *ICML*.

Amos, Brandon, Lei Xu, and J. Zico Kolter (2017). "Input Convex Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 146–155. URL: https://proceedings.mlr.press/v70/amos17b.html.

Sill, Joseph (1997). "Monotonic Networks". In: *Advances in Neural Information Processing Systems*. Ed. by M. Jordan, M. Kearns, and S. Solla. Vol. 10. MIT Press. URL: https://proceedings.neurips.cc/paper/1997/file/83adc9225e4deb67d7ce42d58fe5157c-Paper.pdf.

Um, Kiwon et al. (2020). "Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers". In: *Advances in Neural Information Processing Systems* 33, pp. 6111–6122.

Lange-Hegermann, Markus (2018). "Algorithmic Linearly Constrained Gaussian Processes". In: *NeurIPS*.

Jidling, Carl et al. (2017). "Linearly constrained Gaussian processes". In: *NIPS*.

Hendriks, Johannes N. et al. (2020). "Linearly Constrained Neural Networks". In: *ArXiv* abs/2002.01600.

Lu, Lu et al. (2021b). "Physics-informed neural networks with hard constraints for inverse design". In: *SIAM Journal on Scientific Computing* 43.6, B1105–B1132.

Krantz, Steven George and Harold R Parks (2002). *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media.

Blondel, Mathieu et al. (2021). "Efficient and Modular Implicit Differentiation". In: *ArXiv* abs/2105.15183.

Agrawal, Akshay et al. (2019). "Differentiable Convex Optimization Layers". In: *NeurIPS*.

El Ghaoui, Laurent et al. (2021). "Implicit Deep Learning". In: *SIAM J. Math. Data Sci.* 3, pp. 930–958.

Chen, Tian Qi et al. (2018). "Neural Ordinary Differential Equations". In: *ArXiv* abs/1806.07366.

Donti, Priya L, David Rolnick, and J Zico Kolter (2021). "Dc3: A learning method for optimization with hard constraints". In: *arXiv preprint arXiv:2104.12225*.

Avila Belbute-Peres, Filipe de et al. (2018). "End-to-end differentiable physics for learning and control". In: *Advances in neural information processing systems* 31, pp. 7178–7189.

Pontryagin, Lev Semenovich et al. (1962). "The mathematical theory of optimal processes". In.

Zhang, T. et al. (2019b). *ANODEV2: A Coupled Neural ODE Evolution Framework*. Tech. rep. Preprint: arXiv:1906.04596.

Krishnapriyan, Aditi S et al. (2022). "Learning continuous models for continuous physics". In: *arXiv preprint arXiv:2202.08494*.

Degrave, Jonas et al. (2019). "A Differentiable Physics Engine for Deep Learning in Robotics". In: *Frontiers in Neurorobotics* 13. ISSN: 1662-5218. DOI: 10.3389/fnbot.2019.00006. URL: https://www.frontiersin.org/article/10.3389/fnbot.2019.00006.

Schoenholz, Samuel S. and Ekin D. Cubuk (2020). "JAX M.D. A Framework for Differentiable Physics". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc. URL: https://papers.nips.cc/paper/2020/file/83d3d4b6c9579515e1679aca8cbc8033-Paper.pdf.

Darcy, Henry (1856). *Les fontaines publiques de la ville de Dijon : exposition et application des principes à suivre et des formules à employer dans les questions de distribution d'eau*. Victor Dalmont.

Levenberg, Kenneth (1944). "A METHOD FOR THE SOLUTION OF CERTAIN NON – LINEAR PROBLEMS IN LEAST SQUARES". In: *Quarterly of Applied Mathematics* 2, pp. 164–168.

Marquardt, Donald W. et al. (1963). "An algorithm for least-squares estimation of nonlinear parameters". In.

Hestenes, Magnus R. and Eduard Stiefel (1952). "Methods of conjugate gradients for solving linear systems". In: *Journal of research of the National Bureau of Standards* 49, pp. 409–435.

Saad, Youcef and Martin H. Schultz (1986). "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *Siam Journal on Scientific and Statistical Computing* 7, pp. 856–869.

Driscoll, T. A, N. Hale, and L. N. Trefethen (2014). *Chebfun Guide*. Pafnuty Publications. URL: http://www.chebfun.org/docs/guide/.

Hong, Tao, Pierre Pinson, and Shu Fan (2014). *Global energy forecasting competition 2012*.

Gneiting, Tilmann and Matthias Katzfuss (2014). "Probabilistic Forecasting". In: *Annual Review of Statistics and Its Application* 1.1, pp. 125–151. eprint: https://doi.org/10.1146/annurev-statistics-062713-085831. URL: https://doi.org/10.1146/annurev-statistics-062713-085831.

Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos (2022). "M5 accuracy competition: Results, findings, and conclusions". In: *International Journal of Forecasting* 38.4. Special Issue: M5 competition, pp. 1346–1364. URL: https://www.sciencedirect.com/science/article/pii/S0169207021001874.

Jeon, Jooyoung, Anastasios Panagiotelis, and Fotios Petropoulos (2019). "Probabilistic forecast reconciliation with applications to wind power and electric load". In: *European Journal of Operational Research* 279.2, pp. 364–379.

Rangapuram, Syama Sundar et al. (2021). "End-to-end learning of coherent probabilistic forecasts for hierarchical time series". In: *International Conference on Machine Learning*. PMLR, pp. 8832–8843.

Taieb, Souhaib Ben, James W Taylor, and Rob J Hyndman (2017). "Coherent probabilistic forecasts for hierarchical time series". In: *International conference on machine learning*. PMLR, pp. 3348–3357.

Hyndman, Rob J. et al. (2011). "Optimal combination forecasts for hierarchical time series". In: *Computational Statistics & Data Analysis* 55.9, pp. 2579–2589. URL: https://www.sciencedirect.com/science/article/pii/S0167947311000971.

Wen, Ruofeng et al. (2017). "A Multi-Horizon Quantile Recurrent Forecaster". In: *arXiv: Machine Learning*.

Flunkert, Valentin, David Salinas, and Jan Gasthaus (2017). "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks". In: *ArXiv* abs/1704.04110.

Alexandrov, Alexander et al. (2020). "GluonTS: Probabilistic and Neural Time Series Modeling in Python". In: *Journal of Machine Learning Research* 21.116, pp. 1–6. URL: http://jmlr.org/papers/v21/19-820.html.

Eisenach, Carson, Yagna Patel, and Dhruv Madeka (2020). "MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention". In: *ArXiv* abs/2009.14799.

Benidis, Konstantinos et al. (2022). "Deep learning for time series forecasting: Tutorial and literature survey". In: *ACM Computing Surveys* 55.6, pp. 1–36.

Wang, Bernie et al. (2019). "Deep Factors for Forecasting". In: *International Conference on Machine Learning*.

Kan, Kelvin et al. (2022). "Multivariate quantile function forecaster". In: *AISTATS 2022*. URL: https://www.amazon.science/publications/multivariate-quantile-function-forecaster.

Park, Youngsuk et al. (2022). "Learning Quantile Functions without Quantile Crossing for Distribution-free Time Series Forecasting". In: *ArXiv* abs/2111.06581.

Amos, Brandon and J. Zico Kolter (2017b). "OptNet: Differentiable Optimization as a Layer in Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 136–145. URL: https://proceedings.mlr.press/v70/amos17a.html.

Olivares, Kin G. et al. (2023). "Probabilistic hierarchical forecasting with deep Poisson mixtures". In: *International Journal of Forecasting*. URL: https://www.sciencedirect.com/science/article/pii/S0169207023000432.

Kamarthi, Harshavardhan et al. (2022). "PROFHIT: Probabilistic Robust Forecasting for Hierarchical Time-series". In: *arXiv preprint arXiv:2206.07940*.

Matheson, James E. and Robert L. Winkler (1976). "Scoring Rules for Continuous Probability Distributions". In: *Management Science* 22, pp. 1087–1096.

Legendre, Adrien Marie (1805). *Nouvelles méthodes pour la détermination des orbites de comètes*. F. Didot.

Koenker, Roger and Gilbert Bassett (1978). "Regression Quantiles". In: *Econometrica* 46.1, pp. 33–50. (Visited on 12/30/2022).

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning*.

Hyndman, Rob J and George Athanasopoulos (2013). "Forecasting: principles and practice". In.

Vitullo, Steven R. (2011). "Disaggregating Time Series Data for Energy Consumption by Aggregate and Individual Customer". In.

Hyndman, Rob J., Alan J. Lee, and Earo Wang (2016). "Fast Computation of Reconciled Forecasts for Hierarchical and Grouped Time Series". In: *Comput. Stat. Data Anal.* 97.C, pp. 16–32. URL: https://doi.org/10.1016/j.csda.2015.11.007.

Dangerfield, Byron J. and John S. Morris (1992). "Top-down or bottom-up: Aggregate versus disaggregate extrapolations". In: *International Journal of Forecasting* 8.2, pp. 233–241. URL: https://www.sciencedirect.com/science/article/pii/016920709290121O.

Erven, Tim van and Jairo Cugliari (Dec. 2013). "Game-theoretically Optimal Reconciliation of Contemporaneous Hierarchical Time Series Forecasts". working paper or preprint. URL: https://hal.inria.fr/hal-00920559.

Wickramasuriya, Shanika L., George Athanasopoulos, and Rob J. Hyndman (2019). "Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization". In: *Journal of the American Statistical Association* 114.526, pp. 804–819.

Mishchenko, Konstantin, Mallory Montgomery, and Federico Vaggi (2019). "A Self-supervised Approach to Hierarchical Forecasting with Applications to Groupwise Synthetic Controls". In: *ArXiv* abs/1906.10586.

Han, Xing, Sambarta Dasgupta, and Joydeep Ghosh (2021). "Simultaneously Reconciled Quantile Forecasting of Hierarchically Related Time Series". In: *International Conference on Artificial Intelligence and Statistics*.

Das, Abhimanyu et al. (2022). "A Deep Top-Down Approach to Hierarchically Coherent Probabilistic Forecasting". In.

Taieb, Souhaib Ben and Bonsoo Koo (2019). "Regularized Regression for Hierarchical Forecasting Without Unbiasedness Conditions". In: *KDD*, pp. 1337–1347.

Diaconis, Persi (1977). "Finite forms of de Finetti's theorem on exchangeability". In: *Synthese* 36.2, pp. 271–281.

Diaconis, Persi and David Freedman (1980). "Finite exchangeable sequences". In: *The Annals of Probability*, pp. 745–764.

Zaheer, Manzil et al. (2017). "Deep sets". In: *Advances in neural information processing systems* 30.

Figurnov, Michael, Shakir Mohamed, and Andriy Mnih (2018). "Implicit Reparameterization Gradients". In: *Neural Information Processing Systems*.

Ruiz, Francisco J. R., Michalis K. Titsias, and David M. Blei (2016). "The Generalized Reparameterization Gradient". In: *NIPS*.

Jankowiak, Martin and Fritz Obermeyer (2018). "Pathwise Derivatives Beyond the Reparameterization Trick". In: *ArXiv* abs/1806.01851.

Kingma, Diederik P. and Max Welling (2013). "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114.

Martín Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Gneiting, Tilmann and Adrian E Raftery (2007). "Strictly proper scoring rules, prediction, and estimation". In: *Journal of the American statistical Association* 102.477, pp. 359–378.

Favorita, Corporación et al. (2017). *Corporación Favorita Grocery Sales Forecasting*. URL: https://kaggle.com/competitions/favorita-grocery-sales-forecasting.

Olivares, Kin G. et al. (2022). "HierarchicalForecast: A Reference Framework for Hierarchical Forecasting in Python". In: *Work in progress paper, submitted to Journal of Machine Learning Research*. abs/2207.03517. URL: https://arxiv.org/abs/2207.03517.

Garza, Federico et al. (2022). *StatsForecast: Lightning fast forecasting with statistical and econometric models*. PyCon Salt Lake City, Utah, US 2022. URL: https://github.com/Nixtla/statsforecast.

Hyndman, Rob J and Yeasmin Khandakar (2008). "Automatic time series forecasting: the forecast package for R". In: *Journal of statistical software* 27, pp. 1–22.

Panagiotelis, Anastasios et al. (2023). "Probabilistic forecast reconciliation: Properties, evaluation and score optimisation". In: *European Journal of Operational Research* 306.2, pp. 693–706.

Loshchilov, Ilya and Frank Hutter (2017). "Fixing Weight Decay Regularization in Adam". In: *ArXiv* abs/1711.05101.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980.

Nesterov, Yurii (2013). "Gradient methods for minimizing composite functions". In: *Mathematical Programming*.

Appendix A

Stochastic Frank-Wolfe for constrained finite-sum minimization

A.1 Smoothness

Proposition 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be defined as $f(\boldsymbol{\theta}) = \frac{1}{n}\sum_i f_i(\boldsymbol{\theta}_i)$. If f_i is L-smooth for all $i \in \{1, \dots, n\}$, then f satisfies Eqn. 3.6 for every $p \in [1, \infty]$.*

Proof. We observe that the i-th component of the gradient of f is

$$[\nabla f(\boldsymbol{\theta})]_i = \frac{1}{n} f_i'(\boldsymbol{\theta}_i). \tag{A.1}$$

Recall that $|f_i'(\boldsymbol{\theta}_i) - f_i'(\bar{\boldsymbol{\theta}}_i)| \leq L|\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}}_i|$ for all $\boldsymbol{\theta}_i,\ \bar{\boldsymbol{\theta}}_i$ in the domain of f_i. Then, for the ℓ_p norm $\|\cdot\|_p$ and for all $\boldsymbol{\theta},\ \bar{\boldsymbol{\theta}}$ in the domain of f, the following holds

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\bar{\boldsymbol{\theta}})\|_p = \frac{1}{n}\sqrt[p]{\sum_{i=1}^{n} |f_i'(\boldsymbol{\theta}_i) - f_i'(\bar{\boldsymbol{\theta}}_i)|^p} \leq \frac{L}{n}\sqrt[p]{\sum_{i=1}^{n} |\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}}_i|^p} = \frac{L}{n}\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|_p\ . \tag{A.2}$$

□

A.2 Proof of Lemma 1

We adapt [38]'s proof of Lemma 1. For ease, we reproduce its statement first.

Lemma 4. *Suppose f is a convex function and is (L/n)-smooth with respect to the ℓ_2 norm. For* any *direction $\boldsymbol{\alpha} \in \mathbb{R}^n$, defining $\boldsymbol{s}_t = \mathrm{LMO}(\boldsymbol{X}^\top \boldsymbol{\alpha})$ and $\boldsymbol{w}_t = (1-\gamma_t)\boldsymbol{w}_{t-1} + \gamma_t \boldsymbol{s}_t$, we have the following upper bound on the primal suboptimality*

$$\varepsilon_t \leq \ (1-\gamma_t)\varepsilon_{t-1} + \gamma_t^2 \frac{LD_2^2}{2n} + \gamma_t D_\infty H_t, \tag{A.3}$$

where $\varepsilon_t = f(\boldsymbol{X}\boldsymbol{w}_t) - f(\boldsymbol{X}\boldsymbol{w}_\star)$.

Proof. Recall the definition of $D_p = \max_{\boldsymbol{w},\ \boldsymbol{v}\in\mathcal{C}} \|\boldsymbol{X}(\boldsymbol{w}-\boldsymbol{v})\|_p$.

$$\begin{aligned}
f(\boldsymbol{X}\boldsymbol{w}_t) \leq & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_t - \boldsymbol{w}_{t-1})\rangle + \frac{L}{2n}\|\boldsymbol{X}(\boldsymbol{w}_t - \boldsymbol{w}_{t-1})\|_2^2 && \text{(A.4)}\\
f(\boldsymbol{X}\boldsymbol{w}_t) \leq & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})\rangle + \frac{\gamma_t^2 L}{2n}\|\boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})\|_2^2 && \text{(A.5)}\\
\leq & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})\rangle + \gamma_t^2 \frac{LD_2^2}{2n} && \text{(A.6)}\\
= & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}, \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})\rangle \\
& + \gamma_t\langle \boldsymbol{\alpha}, \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})\rangle + \gamma_t^2 \frac{LD_2^2}{2n} && \text{(A.7)}\\
\leq & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}, \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_\star + \boldsymbol{w}_\star - \boldsymbol{w}_{t-1})\rangle \\
& + \gamma_t\langle \boldsymbol{\alpha}, \boldsymbol{X}(\boldsymbol{w}_\star - \boldsymbol{w}_{t-1})\rangle + \gamma_t^2 \frac{LD_2^2}{2n} && \text{(A.8)}\\
= & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}, \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_\star)\rangle \\
& + \gamma_t\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_\star - \boldsymbol{w}_{t-1})\rangle + \gamma_t^2 \frac{LD_2^2}{2n} && \text{(A.9)}\\
\leq & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_\star - \boldsymbol{w}_{t-1})\rangle \\
& + \gamma_t D_\infty \|\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}\|_1 + \gamma_t^2 \frac{LD_2^2}{2n} && \text{(A.10)}\\
\leq & f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t(f(\boldsymbol{X}\boldsymbol{w}_\star) - f(\boldsymbol{X}\boldsymbol{w}_{t-1})) + \gamma_t D_\infty \|\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}\|_1 \\
& + \gamma_t^2 \frac{LD_2^2}{2n} && \text{(A.11)}
\end{aligned}$$

Subtracting $f(\boldsymbol{X}\boldsymbol{w}_\star)$ on both sides, we get

$$f(\boldsymbol{X}\boldsymbol{w}_t) - f(\boldsymbol{X}\boldsymbol{w}_\star) \leq (1-\gamma_t)(f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - f(\boldsymbol{X}\boldsymbol{w}_\star)) + \gamma_t D_\infty \|\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}\|_1 + \gamma_t^2 \frac{LD_2^2}{2n}. \tag{A.12}$$

We define $H_t = \|\boldsymbol{\alpha} - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1$, and recall the definition of ε_t to obtain the claimed bound

$$\varepsilon_t \leq (1 - \gamma_t)\varepsilon_{t-1} + \gamma_t^2 \frac{LD_2^2}{2n} + \gamma_t D_\infty H_t. \tag{A.13}$$

□

A.3 Completing the proof for Theorem 1

Given the three key Lemmas 1-3, we can finish the proof. Under the hypotheses of Theorem 5, let us consider step t of the SFW algorithm.

We plug our upper bound on H_t Eqn. 3.23 into the upper bound from Lemma 1 Eqn. 3.11 and take expectations on both sides to obtain the following upper bound on the expected primal-suboptimality $\mathbb{E}\varepsilon_t$.

$$\begin{aligned}\mathbb{E}\varepsilon_t \leq & (1-\gamma_t)\mathbb{E}\varepsilon_{t-1} + \gamma_t^2 \frac{LD_2^2}{2n} \\ & + \gamma_t \frac{2LD_1D_\infty}{n}\left(\frac{2(n-1)}{t+2} + \left(1-\frac{1}{n}\right)^{t/2}\log t\right) \\ & + \gamma_t D_\infty \left(1-\frac{1}{n}\right)^t H_0.\end{aligned} \tag{A.14}$$

By specifying the step-size $\gamma_t = \frac{2}{t+2}$ and multiplying the previous inequality by $(t+1)(t+2)$, we get an expression in which the expected sub-optimalities telescope under summation. This allows us to get the promised rate. For simplicity, we upper bound $\frac{t+1}{t+2}$ by 1.

Let $\Gamma_t = (t+1)(t+2)\mathbb{E}\varepsilon_t$. We have

$$\begin{aligned}\Gamma_t \leq & \Gamma_{t-1} + 2\frac{LD_2^2}{n} + 8\frac{(n-1)}{n}LD_1D_\infty \\ & + 4\frac{LD_1D_\infty}{n}(t+1)\Big(1-\frac{1}{n}\Big)^{t/2}\log t \\ & + 2D_\infty H_0(t+1)\left(1-\frac{1}{n}\right)^t\end{aligned} \tag{A.15}$$

If we sum this expression over time-steps $k = 1, \ldots, t$, we obtain

$$\begin{aligned}\Gamma_t \leq & \Gamma_0 + 2L\left(\frac{D_2^2 + 4(n-1)D_1D_\infty}{n}\right)t \\ & + 4\frac{LD_1D_\infty}{n}B_t \\ & + 2D_\infty H_0 C_t\,,\end{aligned} \tag{A.16}$$

where

$$B_t = \sum_{k=1}^{t}(k+1)\left(1-\frac{1}{n}\right)^{k/2}\log k \leq 16n^3 \tag{A.17}$$

$$C_t = \sum_{k=1}^{t}(k+1)\left(1-\frac{1}{n}\right)^{k} \leq n^2\,. \tag{A.18}$$

These bounds use Taylor series and are proven in Appendix A.4. By combining the previous two bounds we get the following upper bound

$$\begin{aligned}\Gamma_t \leq \Gamma_0 + 2L\left(\frac{D_2^2 + 4(n-1)D_1 D_\infty}{n}\right)t \\ + (2D_\infty H_0 + 64LD_1D_\infty)n^2.\end{aligned} \tag{A.19}$$

We divide this upper bound by $(t+1)(t+2)$, and finally use the bound $\frac{1}{(t+1)(t+2)} \leq \frac{1}{t^2}$ to obtain the following rate on $\mathbb{E}\varepsilon_t$:

$$\mathbb{E}\varepsilon_t \leq 2L\left(\frac{D_2^2 + 4(n-1)D_1 D_\infty}{n}\right)\frac{t}{(t+1)(t+2)} + \frac{2\varepsilon_0 + (2D_\infty H_0 + 64LD_1D_\infty)n^2}{(t+1)(t+2)}\ . \tag{A.20}$$

A.4 Bounds for B_t, C_t

For B_t, we use the (aggressive) bound $\log k \leq k-1$ and notice that $\sum_{k=1}^{\infty}(k+2)(k+1)\rho^k = \frac{2}{(1-\rho)^3}$ to get

$$B_t \leq \sum_{k=1}^{t}(k-1)(k+1)\left(1-\frac{1}{n}\right)^{k/2} \tag{A.21}$$

$$\leq \sum_{k=1}^{t}(k+2)(k+1)\left(1-\frac{1}{n}\right)^{k/2} \tag{A.22}$$

$$\leq 2\left(\frac{1}{1-\sqrt{1-\frac{1}{n}}}\right)^3 \tag{A.23}$$

$$= 2n^3\left(1+\sqrt{1-\frac{1}{n}}\right)^3 \leq 16n^3. \tag{A.24}$$

Notice that C_t is the beginning of the Taylor series expansion of $\frac{d}{dx}\frac{1}{1-x} = \frac{1}{(1-x)^2}$, for $x = \frac{n-1}{n}$. We can upper bound it by the full series, leading to

$$C_t \leq \left(\frac{1}{1-\left(1-\frac{1}{n}\right)}\right)^2 = n^2. \tag{A.25}$$

A.5 Convergence of the stochastic gap to the FW gap (Proposition 1.)

Proof. It suffices to prove that

$$|g_t - \hat{g}_t| \leq D_\infty H_t \ . \tag{A.26}$$

We recall the definitions of the true and stochastic FW gaps:

$$g_t = \max_{\boldsymbol{s}\in\mathcal{C}} \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s})\rangle \stackrel{\text{def}}{=} \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s}_t)\rangle \tag{A.27}$$

$$\hat{g}_t = \max_{\boldsymbol{s}\in\mathcal{C}} \langle \boldsymbol{\alpha}_t, \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s})\rangle \stackrel{\text{def}}{=} \langle \boldsymbol{\alpha}_t, \boldsymbol{X}(\boldsymbol{w}_{t-1} - \hat{\boldsymbol{s}}_t)\rangle \tag{A.28}$$

where we associate $\boldsymbol{s}_t$ to the true gap, and $\hat{\boldsymbol{s}}_t$ to the stochastic gap.
Now,

$$\begin{aligned}
g_t &= \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}\left(\boldsymbol{w}_{t-1} - \boldsymbol{s}_t\right)\rangle && \text{(A.29)}\\
&= \langle \boldsymbol{\alpha}_t, \boldsymbol{X}\left(\boldsymbol{w}_{t-1} - \boldsymbol{s}_t\right)\rangle + \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}_t, \boldsymbol{X}\left(\boldsymbol{w}_{t-1} - \boldsymbol{s}_t\right)\rangle && \text{(A.30)}\\
&\leq \langle \boldsymbol{\alpha}_t, \boldsymbol{X}\left(\boldsymbol{w}_{t-1} - \hat{\boldsymbol{s}}_t\right)\rangle + \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}_t, \boldsymbol{X}\left(\boldsymbol{w}_{t-1} - \boldsymbol{s}_t\right)\rangle && \text{(A.31)}\\
&\leq \hat{g}_t + D_\infty H_t, && \text{(A.32)}
\end{aligned}$$

where the first inequality results from optimality of $\hat{\boldsymbol{s}}_t$, and the second inequality results from Hölder's inequality and the definitions of H_t and D_∞.

Both gaps g_t and $\hat{g}_t$ play symmetric roles in the previous bounds, therefore, we also have the bound:

$$\hat{g}_t \leq g_t + D_\infty H_t, \tag{A.33}$$

thus concluding the proof. □

A.6 Proof of Theorem 2

Let us now show that when the f_is are L-smooth, and the iterates are given by the proposed SFW, then $\liminf_{t\to\infty} \mathbb{E}_t[g_t] = 0$.

Proof. We adapt the proof of Lemma 1. At step t, using Proposition 2 with $p = 2$, we obtain

$$f(\boldsymbol{X}\boldsymbol{w}_t) \leq f(\boldsymbol{X}\boldsymbol{w}_{t-1}) + \gamma_t \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})\rangle + \gamma_t^2 \frac{LD_2^2}{2n} \tag{A.34}$$

$$= f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \gamma_t \hat{g}_t + \gamma_t \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}_t, \boldsymbol{X}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})\rangle + \gamma_t^2 \frac{LD_2^2}{2n} \tag{A.35}$$

$$\leq f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - \gamma_t \hat{g}_t + \gamma_t D_\infty H_t + \gamma_t^2 \frac{LD_2^2}{2n} \ . \tag{A.36}$$

Rearranging, we have

$$\gamma_t \hat{g}_t \leq f(\boldsymbol{X}\boldsymbol{w}_{t-1}) - f(\boldsymbol{X}\boldsymbol{w}_t) + \gamma_t D_\infty H_t + \gamma_t^2 \frac{LD_2^2}{2n}. \tag{A.37}$$

Therefore, summing for $u = 1, \ldots, t$

$$\sum_{u=1}^{t} \gamma_u \hat{g}_u \leq f(\boldsymbol{X}\boldsymbol{w}_0) - f(\boldsymbol{X}\boldsymbol{w}_t) + \sum_{u=1}^{t} \gamma_u D_\infty H_u + \gamma_u^2 \frac{LD_2^2}{2n}. \tag{A.38}$$

The right hand side is bounded in expectation: f is continuous on the compact set $\mathcal{C}$, and the series converges, since $\mathbb{E}_t H_t = \mathcal{O}(\frac{1}{t})$ and $\gamma_t^2 = \mathcal{O}(1/t^2)$. This implies that $\liminf \mathbb{E}_t \hat{g}_t = 0$, since $\gamma_t = \frac{2}{t+2}$ is not the general term of a convergent series. Finally, since $|g_t - \hat{g}_t| \leq D_\infty H_t$ (Appendix A.5), this yields the claimed result. □

A.7 Comparison with other methods

To make the comparison with other methods easier to grasp and to implement for the interested reader, we report pseudo code using our notation for the Stochastic Frank-Wolfe algorithms in Lu and Freund [43] and Mokhtari, Hassani, and Karbasi [38]. In the case of Mokhtari, Hassani, and Karbasi [38], we also specify their algorithm in the same formal setting as ours where $f(\boldsymbol{X}\boldsymbol{w}) = \frac{1}{n} f_i(\boldsymbol{x}_i^\top \boldsymbol{w})$ and the sampling is over datapoints.

Mokhtari, Hassani, and Karbasi [38]

Mokhtari, Hassani, and Karbasi [38] have two sets of step-sizes, which we denote by ρ_t, γ_t. They use a form of momentum on an *unbiased* estimator of the gradient using the ρ_t step sizes. The values they use are $\gamma_t = \frac{1}{t+1}$ and $\rho_t = \frac{1}{(t+1)^{2/3}}$.

Algorithm 4 Mokhtari, Hassani, and Karbasi [38]

Initialization: $\boldsymbol{w}_0 \in \mathcal{C}$, $\boldsymbol{\alpha}_0 = 0$, $\mathrm{r}_0 = 0$
for $t = 1, 2, \ldots,$ **do**
  Sample i uniformly at random in $\{1, \ldots, n\}$
  $\boldsymbol{\alpha}_t^i = (1 - \rho_t)\boldsymbol{\alpha}_{t-1}^i + \rho_t f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$
  $\mathrm{r}_t = \mathrm{r}_{t-1} + (\boldsymbol{\alpha}_t^i - \boldsymbol{\alpha}_{t-1}^i)\boldsymbol{x}_i$
  $\boldsymbol{s}_t = \mathrm{LMO}(\mathrm{r}_t)$
  $\boldsymbol{w}_t = (1 - \gamma_t)\boldsymbol{w}_{t-1} + \gamma_t \boldsymbol{s}_t$
end for

Lu and Freund [43]

Lu and Freund [43] also have two step-size sequences given by $\gamma_t = \frac{2(2n_b+t)}{(t+1)(4n_b+t+1)}$ and $\delta_t = \frac{2n_b}{2n_b+t+1}$, where n_b is the number of batches, i.e. $\lfloor n/b \rfloor$, with n the number of samples in the dataset, and b the chosen batch size. They use a form of momentum on the argument to a given f_i, and compute the gradient at an averaged iterate, which we denote by $\boldsymbol{\sigma}_t^i$. In our notation, t is the iteration step and i corresponds to the i-th datapoint.

A.8 Comparison with full-gradient Frank-Wolfe

We derive the rate for the full gradient, deterministic Frank-Wolfe in the finite sum setting, to make comparison with our method easier.

Let us first write the Frank-Wolfe algorithm, using our notations.

Comparison w.r.t. gradient calls.

In the full gradient setting, one iteration makes n gradient calls. Therefore, if $u = nt$ is the number of gradient calls after t iterations, we obtain the following bound in the full gradient case, using the slight abuse of notation ϵ_u to denote the suboptimality after u gradient calls.

$$\varepsilon_u \leq \frac{2LD_2^2}{u}. \tag{A.39}$$

Compare to our method, after t iterations (and therefore after t gradient calls), as shown in Theorem 5

$$\begin{aligned}\mathbb{E}\varepsilon_t \leq & 2L\left(\frac{D_2^2 + 4(n-1)D_1D_\infty}{n}\right)\frac{t}{(t+1)(t+2)} \\ & + \frac{2\varepsilon_0 + (2D_\infty H_0 + 64LD_1D_\infty)n^2}{(t+1)(t+2)}\end{aligned} \tag{A.40}$$

Algorithm 5 Lu and Freund [43]

Initialization: $\boldsymbol{w}_0 \in \mathcal{C}$, $\boldsymbol{\sigma}_0 = \boldsymbol{X}\boldsymbol{w}_0$, $\boldsymbol{\alpha}_0 = 0$, $\mathrm{r}_0 = 0$
for $t = 1, 2, \ldots,$ **do**
  $\boldsymbol{s}_t = \mathrm{LMO}(\mathrm{r}_{t-1})$
  Sample i uniformly at random in $\{1, \ldots, n\}$
  $\boldsymbol{\sigma}_t^i = (1 - \delta_t)\boldsymbol{\sigma}_{t-1}^i + \delta_t(\boldsymbol{x}_i^\top \boldsymbol{s}_t)$
  $\boldsymbol{\alpha}_t^i = \frac{1}{n} f_i'(\boldsymbol{\sigma}_t^i)$
  $\mathrm{r}_t = \mathrm{r}_{t-1} + \left(\boldsymbol{\alpha}_t^i - \boldsymbol{\alpha}_{t-1}^i\right)\boldsymbol{x}_i$
  $\boldsymbol{w}_t = (1 - \gamma_t)\boldsymbol{w}_{t-1} + \gamma_t \boldsymbol{s}_t$
end for

Algorithm 6 Frank-Wolfe algorithm [12]

Initialization: $\boldsymbol{w}_0 \in \mathcal{C}$
for $t = 1, 2, \ldots,$ **do**
  $\boldsymbol{\alpha}_t = \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})$
  $\mathrm{r}_t = \boldsymbol{X}^\top \boldsymbol{\alpha}_t$
  $\boldsymbol{s}_t = \mathrm{LMO}(\mathrm{r}_t)$
  $\boldsymbol{w}_t = (1 - \gamma_t)\boldsymbol{w}_{t-1} + \gamma_t \boldsymbol{s}_t$
end for

Appendix B

Linearly convergent Frank-Wolfe with backtracking line-search

Outline. The supplementary material of this chapter is organized as follows.

- B.1 provides pseudo-code for all FW Variants we consider: AdaFW, AdaAFW, AdaPFW, AdaMP.
- B.2 contains definitions and properties relative to the objective function and/or the domain, such as the definition of geometric strong convexity and pyramidal width.
- B.3 we present key inequalities on the abstract algorithm which are used by the different convergence proofs.
- B.4 provides a proof of convergence for non-convex objectives (Theorem 2).
- B.5 provides a proof of convergence for convex objectives (Theorem 3).
- B.6 provides a proof of linear convergence for all variants except FW (Theorem 4).

B.1 Pseudocode

In this Appendix, we give detailed pseudo-code for the backtracking variants of FW (AdaFW), Away-Steps FW (AdaAFW), Pairwise FW (AdaPFW) and Matching Pursuit (AdaMP).

Backtracking FW

Algorithm 7 Backtracking FW (AdaFW)

Input: $\boldsymbol{x}_0 \in \mathcal{A}$, initial Lipschitz estimate $L_{-1} > 0$, tolerance $\varepsilon \geq 0$, subproblem quality $\delta \in (0, 1]$, adaptivity params $\tau > 1, \eta \geq 1$
for $t = 0, 1 \ldots$ **do**
  Choose any $\boldsymbol{s}_t \in \mathcal{A}$ that satisfies $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle \leq \delta \min_{\boldsymbol{s} \in \mathcal{A}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} - \boldsymbol{x}_t \rangle$
  Set $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{x}_t$ and $\gamma_{\max} = 1$
  Set $g_t = \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle$
  if $g_t \leq \delta\varepsilon$ **then**
    return $\boldsymbol{x}_t$
  end if
  $\gamma_t, L_t = \text{step\_size}(f, \boldsymbol{d}_t, \boldsymbol{x}_t, g_t, L_{t-1}, \gamma_t^{\max})$
  $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$
end for

Backtracking Away-steps FW

Algorithm 8 Backtracking Away-Steps FW (AdaAFW)

Input: $\boldsymbol{x}_0 \in \mathcal{A}$, initial Lipschitz estimate $L_{-1} > 0$, tolerance $\varepsilon \geq 0$, subproblem quality $\delta \in (0, 1]$, adaptivity params $\tau > 1, \eta \geq 1$
Let $\mathcal{S}_0 = \{\boldsymbol{x}_0\}$ and $\boldsymbol{\alpha}_{0,\boldsymbol{v}} = 1$ for $\boldsymbol{v} = \boldsymbol{x}_0$ and $\boldsymbol{\alpha}_{0,\boldsymbol{v}} = 0$ otherwise.
for $t = 0, 1 \ldots$ **do**
  Choose any $\boldsymbol{s}_t \in \mathcal{A}$ that satisfies $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle \leq \delta \min_{\boldsymbol{s} \in \mathcal{A}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} - \boldsymbol{x}_t \rangle$
  Choose any $\boldsymbol{v}_t \in \mathcal{S}_t$ that satisfies $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v}_t \rangle \leq \delta \min_{\boldsymbol{v} \in S_t} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v} \rangle$
  if $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle \leq \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v}_t \rangle$ **then**
    $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{x}_t$
    $\gamma_t^{\max} = 1$
  else
    $\boldsymbol{d}_t = \boldsymbol{x}_t - \boldsymbol{v}_t$, and $\gamma_t^{\max} = \alpha_{\boldsymbol{v}_t,t} / (1 - \alpha_{\boldsymbol{v}_t,t})$
  end if
  Set $g_t = \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle$
  if $g_t \leq \delta\varepsilon$ **then**
    return $\boldsymbol{x}_t$
  end if
  $\gamma_t, L_t = \text{step\_size}(f, \boldsymbol{d}_t, \boldsymbol{x}_t, g_t, L_{t-1}, \gamma_t^{\max})$
  $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$
  Update active set $\mathcal{S}_{t+1}$ and $\boldsymbol{\alpha}_{t+1}$ (see text)
end for

The active set is updated as follows.

- In the case of a FW step, we update the support set $\mathcal{S}_{t+1} = \{\boldsymbol{s}_t\}$ if $\gamma_t = 1$ and otherwise $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{\boldsymbol{s}_t\}$, with coefficients $\alpha_{\boldsymbol{v},t+1} = (1-\gamma_t)\alpha_{\boldsymbol{v},t}$ for $\boldsymbol{v} \in \mathcal{S}_t \setminus \{\boldsymbol{s}_t\}$ and $\alpha_{\boldsymbol{s}_t,t+1} = (1-\gamma_t)\alpha_{\boldsymbol{s}_t,t} + \gamma_t$.
- In the case of an Away step: If $\gamma_t = \gamma_{\max}$, then $\mathcal{S}_{t+1} = \mathcal{S}_t \setminus \{\boldsymbol{v}_t\}$, and if $\gamma_t < \gamma_{\max}$, then $\mathcal{S}_{t+1} = \mathcal{S}_t$. Finally, we update the weights as $\alpha_{\boldsymbol{v},t+1} = (1+\gamma_t)\alpha_{\boldsymbol{v},t}$ for $\boldsymbol{v} \in \mathcal{S}_t \setminus \{\boldsymbol{v}_t\}$ and $\alpha_{\boldsymbol{v}_t,t+1} = (1+\gamma_t)\alpha_{\boldsymbol{v}_t,t} - \gamma_t$ for the other atoms.

Backtracking Pairwise FW

Algorithm 9 Backtracking Pairwise FW (AdaPFW)

Input: $\boldsymbol{x}_0 \in \mathcal{A}$, initial Lipschitz estimate $L_{-1} > 0$, tolerance $\varepsilon \geq 0$, subproblem quality $\delta \in (0, 1]$, adaptivity params $\tau > 1, \eta \geq 1$
Let $\mathcal{S}_0 = \{\boldsymbol{x}_0\}$ and $\boldsymbol{\alpha}_{0,\boldsymbol{v}} = 1$ for $\boldsymbol{v} = \boldsymbol{x}_0$ and $\boldsymbol{\alpha}_{0,\boldsymbol{v}} = 0$ otherwise.
for $t = 0, 1 \ldots$ **do**
  Choose any $\boldsymbol{s}_t \in \mathcal{A}$ that satisfies $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle \leq \delta \min_{\boldsymbol{s} \in \mathcal{A}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} - \boldsymbol{x}_t \rangle$
  Choose any $\boldsymbol{v}_t \in \mathcal{S}_t$ that satisfies $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v}_t \rangle \leq \delta \min_{\boldsymbol{v} \in S_t} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v} \rangle$
  $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{v}_t$ and $\gamma_t^{\max} = \alpha_{\boldsymbol{v}_t,t}$
  Set $g_t = \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle$
  if $g_t \leq \delta\varepsilon$ **then**
    return $\boldsymbol{x}_t$
  end if
  $\gamma_t, L_t = \text{step\_size}(f, \boldsymbol{d}_t, \boldsymbol{x}_t, g_t, L_{t-1}, \gamma_t^{\max})$
  $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$
  Update active set $\mathcal{S}_{t+1}$ and $\boldsymbol{\alpha}_{t+1}$ (see text)
end for

AdaPFW only moves weight from $\boldsymbol{v}_t$ to $\boldsymbol{s}_t$. The active set update becomes $\alpha_{\boldsymbol{s}_t,t+1} = \alpha_{\boldsymbol{s}_t,t} + \gamma_t$, $\alpha_{\boldsymbol{v}_t,t+1} = \alpha_{\boldsymbol{v}_t,t} - \gamma_t$, with $\mathcal{S}_{t+1} = (\mathcal{S}_t \setminus \{\boldsymbol{v}_t\}) \cup \{\boldsymbol{s}_t\}$ if $\alpha_{\boldsymbol{v}_t,t+1} = 0$ and $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{\boldsymbol{s}_t\}$ otherwise.

Backtracking Matching Pursuit

Matching Pursuit [25, 21] is an algorithm to solve optimization problems of the form

$$\underset{\boldsymbol{x} \in \text{lin}(\mathcal{A})}{\text{minimize}} \; f(\boldsymbol{x}) \;, \tag{B.1}$$

where $\text{lin}(\boldsymbol{A}) \stackrel{\text{def}}{=} \left\{ \sum_{\boldsymbol{v} \in \mathcal{A}} \lambda_{\boldsymbol{v}} \boldsymbol{v} \;\middle|\; \lambda_{\boldsymbol{v}} \in \mathbb{R} \right\}$ is the linear span of the set of *atoms* $\mathcal{A}$. As for the backtracking FW algorithm, we assume that f is L-smooth and $\mathcal{A}$ a potentially infinite but bounded set of elements in $\mathbb{R}^p$.

The MP algorithm relies on solving at each iteration a linear subproblem over the set $\mathcal{B} \stackrel{\text{def}}{=} \mathcal{A} \cup -\boldsymbol{A}$, with $-\boldsymbol{A} = \{-\boldsymbol{a} \,|\, \boldsymbol{a} \in \mathcal{A}\}$. The linear subproblem that needs to be solved at each iteration is the following, where as for previous variants, we allow for an optional quality parameter $\delta \in (0, 1]$:

$$\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t \rangle \leq \delta \min_{\boldsymbol{s} \in \mathcal{B}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s} \rangle \ . \tag{B.2}$$

In Algorithm 10 we detail a novel adaptive variant of the MP algorithm, which we name AdaMP.

Algorithm 10 Backtracking Matching Pursuit (AdaMP)

Input: $\boldsymbol{x}_0 \in \mathcal{A}$, initial Lipschitz estimate $L_{-1} > 0$, tolerance $\varepsilon \geq 0$, subproblem quality $\delta \in (0, 1]$
for $t = 0, 1 \ldots$ **do**
  Choose any $\boldsymbol{s}_t \in \mathcal{A}$ that satisfies Eqn. B.2
  $\boldsymbol{d}_t = \boldsymbol{s}_t$
  Set $g_t = \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle$
  if $g_t \leq \delta\varepsilon$ **then**
    return $\boldsymbol{x}_t$
  end if
  $\gamma_t, L_t = \text{step\_size}(f, \boldsymbol{d}_t, \boldsymbol{x}_t, g_t, L_{t-1}, \infty)$
  $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$
end for

B.2 Basic definitions and properties

In this section we give basic definitions and properties relative to the objective function and/or the domain, such as the definition of geometric strong convexity and pyramidal width. These definitions are not specific to our algorithms and have appeared in different sources such as Lacoste-Julien and Jaggi [18] and Locatello et al. [21]. We merely gather them here for completeness.

Definition 1 (Geometric strong convexity)**.** *We define the* ***geometric strong convexity constant*** μ_f^A *as*

$$\mu_f^{\mathcal{A}} \overset{def}{=} \inf_{\substack{\boldsymbol{x},\boldsymbol{x}^\star \in \operatorname{conv}(\mathcal{A}) \\ \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x} \rangle < 0}} \frac{2}{\gamma(\boldsymbol{x}, \boldsymbol{x}^\star)^2} \Big(f(\boldsymbol{x}^\star) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x} \rangle \Big) \tag{B.3}$$

$$\text{where } \gamma(\boldsymbol{x}, \boldsymbol{x}^\star) \overset{def}{=} \frac{\langle -\nabla f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x} \rangle}{\langle -\nabla f(\boldsymbol{x}), \boldsymbol{s}_f(\boldsymbol{x}) - \boldsymbol{v}_f(\boldsymbol{x}) \rangle}, \tag{B.4}$$

where

$$\boldsymbol{s}_f(\boldsymbol{x}) \overset{def}{=} \arg\min_{\boldsymbol{v} \in \mathcal{A}} \langle \nabla f(\boldsymbol{x}), \boldsymbol{v} \rangle \tag{B.5}$$

$$\boldsymbol{v}_f(\boldsymbol{x}) \overset{def}{=} \arg\min_{\substack{\boldsymbol{v} = \boldsymbol{v}_{\mathcal{S}}(\boldsymbol{x}) \\ \mathcal{S} \in \mathcal{S}_{\boldsymbol{x}}}} \langle \nabla f(\boldsymbol{x}), \boldsymbol{v} \rangle \tag{B.6}$$

$$\boldsymbol{v}_{\mathcal{S}}(\boldsymbol{x}) \overset{def}{=} \arg\max_{\boldsymbol{v} \in \mathcal{S}} \langle \nabla f(\boldsymbol{x}), \boldsymbol{v} \rangle \tag{B.7}$$

where $\mathcal{S} \subseteq \boldsymbol{A}$ *and* $\mathcal{S}_{\boldsymbol{x}} \overset{def}{=} \{\mathcal{S} | \mathcal{S} \subseteq \mathcal{A}$ *such that* $\boldsymbol{x}$ *is a proper convex combination of all the elements in* $\mathcal{S}\}$ *(recall* $\boldsymbol{x}$ *is a proper convex combination of elements in* $\mathcal{S}$ *when* $\boldsymbol{x} = \sum_i \alpha_i \boldsymbol{s}_i$ *where* $\boldsymbol{s}_i \in \mathcal{S}$ *and* $\alpha_i \in (0, 1)$*).*

Definition 2 (Pyramidal width)**.** *The* ***pyramidal width*** *of a set* $\boldsymbol{A}$ *is the smallest pyramidal width of all its faces, i.e.*

$$\operatorname{PWidth}(\boldsymbol{A}) \overset{def}{=} \min_{\substack{\mathcal{K} \in \operatorname{faces}(\operatorname{conv}(\boldsymbol{A})) \\ \mathrm{r} \in \operatorname{cone}(\mathcal{K} - \boldsymbol{x}) \setminus \{0\}}} \min_{\boldsymbol{x} \in \mathcal{K}} \operatorname{PdirW}(\mathcal{K} \cap \mathcal{A}, \mathrm{r}, \boldsymbol{x}) \tag{B.8}$$

where PdirW *is the pyramidal directional width, defined as*

$$\operatorname{PdirW}(W)(\boldsymbol{A}, \mathrm{r}, \boldsymbol{x}) \overset{def}{=} \min_{\mathcal{S} \in \mathcal{S}_x} \max_{\boldsymbol{s} \in \boldsymbol{A}, \boldsymbol{v} \in \mathcal{S}} \left\langle \frac{\mathrm{r}}{\|\mathrm{r}\|_2}, \boldsymbol{s} - \boldsymbol{v} \right\rangle \tag{B.9}$$

We now relate these two geometric quantities together.

Lemma 5 (Lower bounding μ_f^A)**.** *Let* f *μ–strongly convex on* $\operatorname{conv}(\mathcal{A}) = \operatorname{conv}(\boldsymbol{A})$*. Then*

$$\mu_f^A \geq \mu \cdot (\operatorname{PWidth}(\boldsymbol{A}))^2 \tag{B.10}$$

Proof. We refer to [18, Theorem 6]. □

Proposition 3. $\mathrm{PWidth}(\boldsymbol{A}) \leq \mathrm{diam}(\mathrm{conv}(\boldsymbol{A}))$ *where* $\mathrm{diam}(\mathcal{X}) \stackrel{def}{=} \sup_{x,y\in\mathcal{X}} \|x-y\|_2$.

Proof. First note that given $\mathrm{r} \in \mathcal{R}$, $\boldsymbol{s} \in \mathcal{S}$, $\boldsymbol{v} \in \mathcal{V}$ with $\mathcal{R}$, $\mathcal{S}$, $\mathcal{V} \subseteq \mathbb{R}^n$, we have

$$\langle \mathrm{r}/\|\mathrm{r}\|_2, \boldsymbol{s}-\boldsymbol{v}\rangle \leq \|\boldsymbol{s}-\boldsymbol{v}\|_2 \qquad \forall \mathrm{r}\in\mathcal{R}, \boldsymbol{s}\in\mathcal{S}, \boldsymbol{v}\in\mathcal{V} \tag{B.11}$$

$$\Rightarrow \max_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{v}\in\mathcal{V}} \langle \mathrm{r}/\|\mathrm{r}\|_2, \boldsymbol{s}-\boldsymbol{v}\rangle \leq \max_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{v}\in\mathcal{V}} \|\boldsymbol{s}-\boldsymbol{v}\|_2 \qquad \forall \mathrm{r}\in\mathcal{R} \tag{B.12}$$

$$\Rightarrow \min_{\mathrm{r}\in\mathcal{R}} \max_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{v}\in\mathcal{V}} \langle \mathrm{r}/\|\mathrm{r}\|_2, \boldsymbol{s}-\boldsymbol{v}\rangle \leq \max_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{v}\in\mathcal{V}} \|\boldsymbol{s}-\boldsymbol{v}\|_2 \tag{B.13}$$

Applying this result to the definition of pyramidal width we have

$$\mathrm{PWidth}(\boldsymbol{A}) = \min_{\substack{\boldsymbol{x}\in\mathcal{K} \\ \mathcal{K}\in\mathrm{faces}(\mathrm{conv}(\boldsymbol{A})) \\ \mathrm{r}\in\mathrm{cone}(\mathcal{K}-\boldsymbol{x})\setminus\{0\}}} \mathrm{PdirW}(\mathcal{K}\cap\mathcal{A}, \mathrm{r}, \boldsymbol{x}) \tag{B.14}$$

$$= \min_{\substack{\boldsymbol{x}\in\mathcal{K} \\ \mathcal{K}\in\mathrm{faces}(\mathrm{conv}(\boldsymbol{A})) \\ \mathrm{r}\in\mathrm{cone}(\mathcal{K}-\boldsymbol{x})\setminus\{0\}}} \min_{\mathcal{S}\in\mathcal{S}_x} \max_{\boldsymbol{s}\in\boldsymbol{A},\boldsymbol{v}\in\mathcal{S}} \left\langle \frac{\mathrm{r}}{\|\mathrm{r}\|}, \boldsymbol{s}-\boldsymbol{v} \right\rangle \tag{B.15}$$

$$= \min_{r\in\mathcal{R}} \max_{\boldsymbol{s}\in\boldsymbol{A},v\in\mathcal{V}} \left\langle \frac{\mathrm{r}}{\|\mathrm{r}\|}, \boldsymbol{s}-\boldsymbol{v} \right\rangle \tag{B.16}$$

$$\tag{B.17}$$

where $\mathcal{R} = \{\mathrm{cone}(\mathcal{K}-\boldsymbol{x})\setminus\{0\} : \text{for some } \boldsymbol{x}\in\mathcal{K},\ \mathcal{K}\in\mathrm{faces}(\mathrm{conv}(\boldsymbol{A}))\}$ and $\mathcal{V}$ is some subset of $\boldsymbol{A}$. Applying the derived result we have that

$$\begin{aligned}\mathrm{PWidth}(\boldsymbol{A}) &\leq \max_{\boldsymbol{s}\in\boldsymbol{A},\boldsymbol{v}\in\mathcal{V}} \|\boldsymbol{s}-\boldsymbol{v}\|_2 \\ &\leq \max_{\boldsymbol{s},\boldsymbol{v}\in\mathrm{conv}(\boldsymbol{A})} \|\boldsymbol{s}-\boldsymbol{v}\|_2 \\ &= \mathrm{diam}(\mathrm{conv}(\boldsymbol{A}))\end{aligned}$$

□

Definition 3. *The* ***minimal directional width*** $\mathrm{mDW}(\boldsymbol{A})$ *of a set of atoms* $\boldsymbol{A}$ *is defined as*

$$\mathrm{mDW}(\boldsymbol{A}) = \min_{\boldsymbol{d}\in\mathrm{lin}(\boldsymbol{A})} \max_{\boldsymbol{z}\in\boldsymbol{A}} \frac{\langle \boldsymbol{z}, \boldsymbol{d}\rangle}{\|\boldsymbol{d}\|}\ . \tag{B.18}$$

Note that in contrast to the pyramidal width, the minimal directional width here is a much simpler and robust property of the atom set $\boldsymbol{A}$, not depending on its combinatorial face structure of the polytope. As can be seen directly from the definition above, the $\mathrm{mDW}(\boldsymbol{A})$ is robust when adding a duplicate atom or small perturbation of it to $\boldsymbol{A}$.

B.3 Preliminaries: Key Inequalities

In this appendix we prove that the sufficient decrease condition verifies a recursive inequality. This key result is used by all convergence proofs.

Lemma 6. *The following inequality is verified for all proposed algorithms (with $\gamma_t^{\max} = +\infty$ for AdaMP):*

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \xi g_t + \frac{\xi^2 L_t}{2}\|\boldsymbol{d}_t\|^2 \quad \textit{for all } \xi \in [0, \gamma_t^{\max}]. \tag{B.19}$$

Proof. We start the proof by proving an optimality condition of the step-size. Consider the following quadratic optimization problem:

$$\underset{\xi \in [0, \gamma_t^{\max}]}{\text{minimize}} \; -\xi g_t + \frac{L_t \xi^2}{2}\|\boldsymbol{d}_t\|^2 \; . \tag{B.20}$$

Deriving with respect to ξ and noting that on all the considered algorithms we have $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle \leq 0$, one can easily verify that the global minimizer is achieved at the value

$$\min\left\{ \frac{g_t}{L_t \|\boldsymbol{d}_t\|^2}, \gamma_t^{\max} \right\} \; , \tag{B.21}$$

where $g_t = \langle -\nabla f(\boldsymbol{x}), \boldsymbol{d}_t \rangle$. This coincides with the value of γ_{t+1} computed by the backtracking procedure on the different algorithms and so we have:

$$-\gamma_t g_t + \frac{L_t {\gamma_t}^2}{2}\|\boldsymbol{d}_t\|^2 \leq -\xi g_t + \frac{L_t \xi^2}{2}\|\boldsymbol{d}_t\|^2 \;\; \text{for all } \xi \in [0, \gamma^{\max}] \; . \tag{B.22}$$

We can now write the following sequence of inequalities, that combines the sufficient decrease condition with this last inequality:

$$f(\boldsymbol{x}_{t+1}) \;\leq\; f(\boldsymbol{x}_t) - \gamma_t g_t + \frac{L_t \gamma_t^2}{2}\|\boldsymbol{d}_t\|^2 \tag{B.23}$$

$$\overset{\textit{Eqn. B.20}}{\leq} \; f(\boldsymbol{x}_t) - \xi g_t + \frac{L_t \xi^2}{2}\|\boldsymbol{d}_t\|^2 \;\; \text{for any } \xi \in [0, \gamma^{\max}] \; . \tag{B.24}$$

□

Proposition 4. *The Lipschitz estimate L_t is bounded as $L_t \leq \max\{\tau L, L_{-1}\}$.*

Proof. If the sufficient decrease condition is verified then we have $L_t = \eta L_{t-1}$ and so $L_t \leq L_{t-1}$. If it's not, we at least have that the Lipschitz estimate cannot larger than τL by definition of Lipschitz constant. Combining both bounds we obtain

$$L_t \leq \max\{\tau L, L_{t-1}\} \; . \tag{B.25}$$

Applying the same bound recursively on L_{t-1} leads to the claimed bound $L_t \leq \max\{\tau L, L_{-1}\}$.
□

Lemma 7. *Let $g(\cdot)$ be as in Theorem 2, i.e., $g(\cdot) = g^{FW}(\cdot)$ for FW variants (AdaFW, AdaAFW, AdaPFW) and $g(\cdot) = g^{MP}(\cdot)$ for MP variants (AdaMP). Then for any of these algorithms we have*

$$g_t \geq \delta g(\boldsymbol{x}_t) \ . \tag{B.26}$$

Proof. • For AdaFW and AdaMP, Eq. Eqn. B.26 follows immediately from the definition of g_t and $g(\boldsymbol{x}_t)$.

• For AdaAFW, by the way the descent direction is selected in Line 6, we always have

$$g_t \geq \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s}_t \rangle \geq \delta g(\boldsymbol{x}_t) \ , \tag{B.27}$$

where the last inequality follows from the definition of $\boldsymbol{s}_t$

• For AdaPFW, we have

$$\begin{aligned} g_t &= \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{v}_t - \boldsymbol{s}_t \rangle = \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s}_t \rangle + \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{v}_t - \boldsymbol{x}_t \rangle & \text{(B.28)} \\ &\geq \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s}_t \rangle \geq \delta g(\boldsymbol{x}_t) & \text{(B.29)} \end{aligned}$$

where the term $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{v}_t - \boldsymbol{x}_t \rangle$ is positive by definition of $\boldsymbol{v}_t$ since $\boldsymbol{x}_t$ is necessarily in the convex envelope of $\mathcal{S}_t$. The second inequality follows from the definition of $\boldsymbol{s}_t$.

□

Theorem 1. *Let N_t be the total number of evaluations of the sufficient decrease condition up to iteration t. Then we have*

$$n_t \leq \left[1 - \frac{\log \eta}{\log \tau}\right](t+1) + \frac{1}{\log \tau} \max\left\{\log \frac{\tau L}{L_{-1}}, 0\right\} \ . \tag{B.30}$$

Proof. This proof follows roughly that of [144, Lemma 3], albeit with a slightly different bound on L_t due to algorithmic differences.

Denote by $n_i \geq 1$ the number of evaluations of the sufficient decrease condition. Since the algorithm multiplies by τ every time that the sufficient condition is not verified, we have

$$L_i = \eta L_{i-1} \tau^{n_i - 1} \ . \tag{B.31}$$

Taking logarithms on both sides we obtain

$$n_i \leq 1 - \frac{\log \eta}{\log \tau} + \frac{1}{\tau} \log \frac{L_i}{L_{i-1}} \ . \tag{B.32}$$

Summing from $i = 0$ to $i = t$ gives

$$n_t \leq \sum_{i=0}^{t} n_i = \left[1 - \frac{\log \eta}{\log \tau}\right](t+1) + \frac{1}{\log \tau} \log\left(\frac{L_t}{L_{-1}}\right) \tag{B.33}$$

Finally, from Proposition 4 we have the bound $L_t \leq \max\{\tau L, L_{-1}\}$, which we can use to bound the numerator's last term. This gives the claimed bound

$$n_t \leq \sum_{i=0}^{t} n_i = \left[1 - \frac{\log \eta}{\log \tau}\right](t+1) + \frac{1}{\log \tau} \max\left\{\log \frac{\tau L}{L_{-1}}, 0\right\} \ . \tag{B.34}$$

□

A bound on the number of bad steps

To prove the linear rates for the backtracking AFW and backtracking PFW algorithm it is necessary to bound the number of bad steps. There are two different types of bad steps: "drop" steps and "swap" steps. These names come from how the active set $\mathcal{S}_t$ changes. In a drop step, an atom is removed from the active set (i.e. $|\mathcal{S}_{t+1}| < |\mathcal{S}_t|$). In a swap step, the size of the active set remains unchanged (i.e. $|\mathcal{S}_{t+1}| = |\mathcal{S}_t|$) but one atom is swapped with another one not in the active set. Note that drop steps can occur in the (backtracking) Away-steps and Pairwise, but swap steps can only occur in the Pairwise variant.

For the proofs of linear convergence in B.6, we show that these two types of bad steps are only problematic when $\gamma_t = \gamma_t^{\max} < 1$. In these scenarios, we cannot provide a meaningful decrease bound. However, we show that the number of bad steps we take is bounded. The following two lemmas adopted from [18, Appendix C] bound the number of drop steps and swap steps the backtracking algorithms can take.

Lemma 8. *After T steps of AdaAFW or AdaPFW, there can only be $T/2$ drop steps. Also, if there is a drop step at step $t+1$, then $f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}_t) < 0$.*

Proof. Let A_t denote the number of steps that added a vertex in the expansion, and let D_t be the number of drop steps. Then $1 \leq |\mathcal{S}_t| = |\mathcal{S}_0| + A_t - D_t$ and we clearly have $A_t - D_t \leq t$. Combining these two inequalities we have that $D_t \leq \frac{1}{2}(|\mathcal{S}_0| - 1 + t) = \frac{t}{2}$.

To show $f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}_t) < 0$, because of Lemma 6, it suffices to show that

$$-\gamma_t g_t + \frac{1}{2}\gamma_t^2 L_t \|\boldsymbol{d}_t\|^2 < 0 \ , \tag{B.35}$$

with $\gamma_t = \gamma_t^{\max}$ (recall drop steps only occur when $\gamma_t = \gamma_t^{\max}$). Note this is a convex quadratic in γ_t which is precisely less than or equal to 0 when $\gamma_t \in [0, 2g_t/L_t\|\boldsymbol{d}_t\|^2]$. Thus in order to show $f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}_t) < 0$ it suffices to show $\gamma_t^{\max} \in (0, 2g_t/L_t\|\boldsymbol{d}_t\|^2)$. This follows immediately since $0 < \gamma_t^{\max} \leq g_t/L_t\|\boldsymbol{d}_t\|^2$. □

Since in the AdaAFW algorithm all bad steps are drop steps, the previous lemma implies that we can effectively bound the number of bad steps by $t/2$, which is the bound claimed in Eqn. 2.7.

Lemma 9. *There are at most $3|\boldsymbol{A}|!$ bad steps between any two good steps in AdaPFW. Also, if there is a swap step at step $t+1$, then $f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}_t) < 0$.*

Proof. Note that bad steps only occur when $\gamma_t = \gamma_t^{\max} = \alpha_{\boldsymbol{v}_t,t}$. When this happens there are two possibilities; we either move all the mass from $\boldsymbol{v}_t$ to a new atom $\boldsymbol{s}_t \notin \mathcal{S}_t$ (i.e. $\alpha_{\boldsymbol{v}_t,t+1} = 0$ and $\alpha_{\boldsymbol{s}_t,t+1} = \alpha_{\boldsymbol{v}_t,t}$) and preserve the cardinality of our active set ($|\mathcal{S}_{t+1}| = |\mathcal{S}_t|$) or we move all the mass from $\boldsymbol{v}_t$ to an old atom $\boldsymbol{s}_t \in \mathcal{S}_t$ (i.e. $\alpha_{\boldsymbol{s}_t,t+1} = \alpha_{\boldsymbol{s}_t,t} + \alpha_{\boldsymbol{v}_t,t}$) and the cardinality of our active set decreases by 1 ($|\mathcal{S}_{t+1}| < |\mathcal{S}_t|$). In the former case, the possible values of the coordinates $\alpha_{\boldsymbol{v}}$ do not change, but they are simply rearranged in the possible $|\boldsymbol{A}|$ slots. Note further every time the mass from $\boldsymbol{v}_t$ moves to a new atom $\boldsymbol{s}_t \notin \mathcal{S}_t$ we have strict descent, i.e. $f(\boldsymbol{x}_{t+1}) < f(\boldsymbol{x}_t)$ unless $\boldsymbol{x}_t$ is already optimal (see Lemma 8) and hence we cannot revisit the

same point unless we have converged. Thus the maximum number of possible consecutive swap steps is bounded by the number of ways we can assign $|\mathcal{S}_t|$ numbers in $|\boldsymbol{A}|$ slots, which is $|\boldsymbol{A}|!/(|\boldsymbol{A}| - |\mathcal{S}_t|)!$. Furthermore, when the cardinality of our active set drops, in the worst case we will do a maximum number of drop steps before reducing the cardinality of our active set again. Thus starting with $|\mathcal{S}_t| = r$ the maximum number of bad steps B without making any good steps is upper bounded by

$$B \leq \sum_{k=1}^{r} \frac{|\boldsymbol{A}|!}{(|\boldsymbol{A}| - k)!} \leq |\boldsymbol{A}|! \sum_{k=0}^{\infty} \frac{1}{k!} = |\boldsymbol{A}|! e \leq 3|\boldsymbol{A}|!$$

□

B.4 Proofs of convergence for non-convex objectives

In this appendix we provide the convergence proof of Theorem 2. Although this theorem provides a unified convergence proof for both variants of FW and MP, for convenience we split the proof into one for FW variants (Theorem 2.A) and another one for variants of MP (Theorem 2.B)

Theorem 2.A. *Let $\boldsymbol{x}_t$ denote the iterate generated by either AdaFW, AdaAFW or AdaPFW after t iterations. Then for any iteration t with $N_{t+1} \geq 0$, we have the following suboptimality bound in terms of the FW gap:*

$$\lim_{k\to\infty} g^{FW}(\boldsymbol{x}_k) = 0 \quad \textit{and} \quad \min_{k=0,\dots,t} g^{FW}(\boldsymbol{x}_k) \leq \frac{\max\{2h_0, L_t^{\max}\operatorname{diam}(\mathcal{A})^2\}}{\delta\sqrt{N_{t+1}}} = \mathcal{O}\left(\frac{1}{\delta\sqrt{t}}\right) \tag{B.36}$$

Proof. By Lemma 6 we have the following inequality for any k and any $\xi \in [0, \gamma_k^{\max}]$,

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \xi g_k + \frac{\xi^2 C_k}{2} \ , \tag{B.37}$$

where we define $C_k \stackrel{\text{def}}{=} L_k\|\boldsymbol{d}_k\|^2$ for convenience. We consider now different cases according to the relative values of γ_k and $\gamma_k^{\max}$, yielding different upper bounds for the right hand side.

Case 1: $\gamma_k < \gamma_k^{\max}$
In this case, γ_k maximizes the right hand side of the (unconstrained) quadratic in inequality Eqn. B.37 which then becomes:

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \frac{g_k^2}{2C_k} \leq f(\boldsymbol{x}_k) - \frac{g_k}{2}\min\left\{\frac{g_k}{C_k}, 1\right\} \tag{B.38}$$

Case 2: $\gamma_k = \gamma_k^{\max} \geq 1$
By the definition of γ_t, this case implies that $C_k \leq g_k$ and so using $\xi = 1$ in Eqn. B.37 gives

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -g_k + \frac{C_k}{2} \leq -\frac{g_k}{2} \ . \tag{B.39}$$

Case 3: $\gamma_k = \gamma_k^{\max} < 1$
This corresponds to the problematic drop steps for AdaAFW or possibly swap steps for AdaPFW, in which we will only be able to guarantee that the iterates are non-increasing. Choosing $\xi = 0$ in Eqn. B.37 we can at least guarantee that the objective function is non-increasing:

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq 0 \ . \tag{B.40}$$

Combining the previous cases. We can combine the inequalities obtained for the previous cases into the following inequality, valid for all $k \leq t$,

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\frac{g_k}{2} \min\left\{\frac{g_k}{C_k}, 1\right\} \mathbb{1}\{k \text{ is a good step}\} \tag{B.41}$$

Adding the previous inequality from $k = 0$ up to t and rearranging we obtain

$$f(\boldsymbol{x}_0) - f(\boldsymbol{x}_{t+1}) \geq \sum_{k=0}^{t} \frac{g_k}{2} \min\left\{\frac{g_k}{L_k\|\boldsymbol{d}_k\|^2}, 1\right\} \mathbb{1}\{k \text{ is a good step}\} \tag{B.42}$$

$$\geq \sum_{k=0}^{t} \frac{g_k}{2} \min\left\{\frac{g_k}{C_k^{\max}}, 1\right\} \mathbb{1}\{k \text{ is a good step}\} \tag{B.43}$$

with $C_t^{\max} \stackrel{\text{def}}{=} L_t^{\max} \operatorname{diam}(\operatorname{conv}(\mathcal{A}))^2$. Taking the limit for $t \to +\infty$ we obtain that the left hand side is bounded by the compactness assumption on the domain $\operatorname{conv}(\mathcal{A})$ and L-smoothness on f. The right hand side is an infinite sum, and so a necessary condition for it to be bounded is that $g_k \to 0$, since $g_k \geq 0$ for all k. We have hence proven that $\lim_{k\to\infty} g_k = 0$, which by Lemma 7 implies $\lim_{k\to\infty} g(\boldsymbol{x}_k) = 0$. This proves the first claim of the Theorem.

We will now aim to derive explicit convergence rates for convergence towards a stationary point. Let $\widetilde{g}_t = \min_{0\leq k\leq t} g_k$, then from Eq. Eqn. B.43 we have

$$f(\boldsymbol{x}_0) - f(\boldsymbol{x}_{t+1}) \geq \sum_{k=0}^{t} \frac{\widetilde{g}_t}{2} \min\left\{\frac{\widetilde{g}_t}{C_t^{\max}}, 1\right\} \mathbb{1}\{k \text{ is a good step}\} \tag{B.44}$$

$$= N_{t+1} \frac{\widetilde{g}_t}{2} \min\left\{\frac{\widetilde{g}_t}{C_t^{\max}}, 1\right\} \ . \tag{B.45}$$

We now make a distinction of cases for the quantities inside the min.

- If $\widetilde{g}_t \leq C_t^{\max}$, then Eqn. B.45 gives $f(\boldsymbol{x}_0) - f(\boldsymbol{x}_{t+1}) \geq N_{t+1}\widetilde{g}_t^2/(2C_t^{\max})$, which reordering gives

$$\widetilde{g}_t \leq \sqrt{\frac{2C_t^{\max}(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_{t+1}))}{N_{t+1}}} \leq \sqrt{\frac{2C_t^{\max}h_0}{N_{t+1}}} \leq \frac{2h_0 + C_t^{\max}}{2\sqrt{N_{t+1}}} \leq \frac{\max\{2h_0, C_t^{\max}\}}{\sqrt{N_{t+1}}} \ . \tag{B.46}$$

  where in the third inequality we have used the inequality $\sqrt{ab} \leq \frac{a+b}{2}$ with $a = \sqrt{2h_0}$, $b = \sqrt{C_t^{\max}}$.

- If $\widetilde{g}_t > C_t^{\max}$ we can get a better $\frac{1}{N_t}$ rate, trivially bounded by $\frac{1}{\sqrt{N_t}}$.

$$\widetilde{g}_t \leq \frac{2h_0}{N_{t+1}} \leq \frac{2h_0}{\sqrt{N_{t+1}}} \leq \frac{\max\{2h_0, C_t^{\max}\}}{\sqrt{N_{t+1}}} \ . \tag{B.47}$$

We have obtained the same bound in both cases, hence we always have

$$\widetilde{g}_t \leq \frac{\max\{2h_0, C_t^{\max}\}}{\sqrt{N_{t+1}}} \ . \tag{B.48}$$

Finally, from Lemma 7 we have $g(\boldsymbol{x}_k) \leq \frac{1}{\delta} g_k$ for all k and so

$$\min_{0\leq k\leq t} g(\boldsymbol{x}_k) \leq \frac{1}{\delta} \min_{0\leq k\leq t} g_k = \frac{1}{\delta}\widetilde{g}_t \leq \frac{\max\{2h_0, C_t^{\max}\}}{\delta\sqrt{N_{t+1}}} \ , \tag{B.49}$$

and the claimed bound follows by definition of $C_t^{\max}$. The $\mathcal{O}(1/\delta\sqrt{t})$ rate comes from the fact that both $\overline{L}_t$ and h_0 are upper bounded. $\overline{L}_t$ is bounded by Proposition 4 and h_0 is bounded by assumption.

□

Matching Pursuit

In the context of Matching Pursuit, we propose the following criterion which we name the MP gap: $g^{\text{MP}}(\boldsymbol{x}) = \max_{\boldsymbol{s}\in\mathcal{B}}\langle\nabla f(\boldsymbol{x}), \boldsymbol{s}\rangle$, where $\mathcal{B}$ is as defined in B.1. Note that g^{MP} is always non-negative and $g^{\text{MP}}(\boldsymbol{x}^\star) = 0$ implies $\langle\nabla f(\boldsymbol{x}^\star), \boldsymbol{s}\rangle = 0$ for all $\boldsymbol{s} \in \mathcal{B}$. By linearity of the inner product we then have $\langle\nabla f(\boldsymbol{x}^\star), \boldsymbol{x} - \boldsymbol{x}^\star\rangle = 0$ for any $\boldsymbol{x}$ in the domain, since $\boldsymbol{x} - \boldsymbol{x}^\star$ lies in the linear span of $\mathcal{A}$. Hence $\boldsymbol{x}^\star$ is a stationary point and g^{MP} is an appropriate measure of stationarity for this problem.

Theorem 2.B. *Let $\boldsymbol{x}_t$ denote the iterate generated by AdaMP after t iterations. Then for $t \geq 0$ we have the following suboptimality bound in terms of the MP gap:*

$$\lim_{k\to\infty} g^{MP}(\boldsymbol{x}_k) = 0 \qquad \textit{and} \qquad \min_{0\leq k\leq t} g^{MP}(\boldsymbol{x}_k) \leq \frac{\text{radius}(\boldsymbol{A})}{\delta}\sqrt{\frac{2h_0\overline{L}_t}{t+1}} = \mathcal{O}\left(\frac{1}{\delta\sqrt{t}}\right) \ . \tag{B.50}$$

Proof. The proof similar than that of Theorem 2.A, except that in this case the expression of the step-size is simpler and does not depend on the minimum of two quantities. This avoids the case distinction that was necessary in the previous proof, resulting in a much simpler proof.

For all $k = 0, \dots, t$, using the sufficient decrease condition, and the definitions of γ_k and g_k:

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq \gamma_k \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k \rangle + \frac{\gamma_k^2 L_k}{2} \|\boldsymbol{d}_k\|^2 \tag{B.51}$$

$$\leq \min_{\eta \geq 0} \left\{ -\eta g_k + \frac{1}{2} \eta^2 L_k \|\boldsymbol{d}_k\|^2 \right\} \tag{B.52}$$

$$\leq -\frac{g_k^2}{2L_k \|\boldsymbol{d}_k\|^2} \ , \tag{B.53}$$

where the last inequality comes from minimizing with respect to η. Summating over k from 0 to t and negating the previous inequality, we obtain:

$$\sum_{0 \leq k \leq t} \frac{g_k^2}{L_k} \leq (f(\boldsymbol{x}_0) - f(\boldsymbol{x}_t)) \operatorname{radius}(\boldsymbol{A})^2 \leq 2h_0 \operatorname{radius}(\boldsymbol{A})^2 \ . \tag{B.54}$$

Taking the limit for $t \to \infty$ we obtain that the left hand side has a finite sum since the right hand side is bounded by assumption. Therefore, $g_k \to 0$, which by Lemma 7 implies $\lim_{k \to \infty} g(\boldsymbol{x}_k) = 0$. This proves the first claim of the Theorem.

We now aim to derive explicit convergence rates. Taking the min over the g_ks and taking a square root for the last inequality

$$\min_{0 \leq k \leq t} g_k \leq \sqrt{\frac{2h_0 \operatorname{radius}(\boldsymbol{A})^2}{\sum_{0 \leq k \leq t} {L_k}^{-1}}} \tag{B.55}$$

The term $\left(n / \sum_{0 \leq k \leq t} {L_k}^{-1}\right)$ is the *harmonic mean* of the L_ks, which is always upper bounded by the average $\overline{L}_t$. Hence we obtain

$$\min_{0 \leq k \leq t} g_k \leq \frac{\operatorname{radius}(\boldsymbol{A})}{\delta} \sqrt{\frac{2h_0 \overline{L}_t}{t+1}} \ . \tag{B.56}$$

The claimed rate then follows from using the bound $g(\boldsymbol{x}_k) \leq \frac{1}{\delta} g_k$ from Lemma 7, valid for all $k \geq 0$.

The $\mathcal{O}(1/\delta\sqrt{t})$ rate comes from the fact that both $\overline{L}_t$ and h_0 are upper bounded. $\overline{L}_t$ is bounded by Proposition 4 and h_0 is bounded by assumption.

□

Note: Harmonic mean vs arithmetic mean. The convergence rate for MP on non-convex objectives (Theorem 2) also holds by replacing $\overline{L}_t$ by its harmonic mean $H_t \stackrel{\text{def}}{=} N_t / (\sum_{k=0}^{t-1} L_k^{-1} \mathbb{1}\{k \text{ is a good step}\})$ respectively. The harmonic mean is always less than the arithmetic mean, i.e., $H_t \leq \overline{L}_t$, although for simplicity we only stated both theorems with the arithmetic mean. Note that the Harmonic mean is Schur-concave, implying that $H_t \leq t \min\{L_k : k \leq t\}$, i.e. it is controlled by the smallest Lipschitz estimate encountered so far.

B.5 Proofs of convergence for convex objectives

In this section we provide a proof the convergence rates stated in the theorem for convex objectives (Theorem 3). The section is structured as follows. We start by proving a technical result which is a slight variation of Lemma 6 and which will be used in the proof of Theorem 3. This is followed by the proof of Theorem 3.

Frank-Wolfe variants

Lemma 10. *For any of the proposed FW variants, if t is a good step, then we have*

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \xi g_t + \frac{\xi^2 L_t}{2}\|\boldsymbol{d}_t\|^2 \quad \textit{for all } \xi \in [0,1]. \tag{B.57}$$

Proof. If $\gamma_t^{\max} \geq 1$, the result is obvious from Lemma 6. If $\gamma_t^{\max} < 1$, then the inequality is only valid in the smaller interval $[0, \gamma_t^{\max}]$. However, since we have assumed that this is a good step, if $\gamma_t^{\max} < 1$ then we must have $\gamma_t < \gamma_t^{\max}$. By Lemma 6, we have

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) + \min_{\xi \in [0, \gamma_t^{\max}]} \left\{ \xi \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \xi^2}{2}\|\boldsymbol{d}_t\|^2 \right\} \tag{B.58}$$

Because $\gamma_t < \gamma_t^{\max}$ and since the expression inside the minimization term of the previous equation is a quadratic function of ξ, γ_t is the unconstrained minimum and so we have

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) + \min_{\xi \geq 0} \left\{ \xi \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \xi^2}{2}\|\boldsymbol{d}_t\|^2 \right\} \tag{B.59}$$

$$\leq f(\boldsymbol{x}_t) + \min_{\xi \in [0,1]} \left\{ \xi \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \xi^2}{2}\|\boldsymbol{d}_t\|^2 \right\} . \tag{B.60}$$

The claimed bound then follows from the optimality of the min. □

The following lemma allows to relate the quantity $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s}_t \rangle$ with a primal-dual gap and will be essential in the proof of Theorem 3.

Lemma 11. *Let $\boldsymbol{s}_t$ be as defined in any of the FW variants. Then for any iterate $t \geq 0$ we have*

$$\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s}_t \rangle \geq \delta(f(\boldsymbol{x}_t) - \psi(\nabla f(\boldsymbol{x}_t))) . \tag{B.61}$$

Proof.

$$\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s}_t \rangle \overset{\textit{Eqn. 2.2}}{\geq} \delta \max_{\boldsymbol{s} \in \operatorname{conv}(\mathcal{A})} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s} \rangle \tag{B.62}$$

$$= \delta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t \rangle + \delta \max_{\boldsymbol{s} \in \operatorname{conv}(\mathcal{A})} \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s} \rangle \tag{B.63}$$

$$= \delta\big(\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t \rangle + \sigma_{\operatorname{conv}(\mathcal{A})}(-\nabla f(\boldsymbol{x}_t)) \big) \tag{B.64}$$

$$= \delta\big(f(\boldsymbol{x}_t) + \underbrace{f^*(\nabla f(\boldsymbol{x}_t)) + \sigma_{\operatorname{conv}(\mathcal{A})}(-\nabla f(\boldsymbol{x}_t))}_{= -\psi(\nabla f(\boldsymbol{x}_t))} \big) = \delta\big(f(\boldsymbol{x}_t) - \psi(\nabla f(\boldsymbol{x}_t)) \big) \tag{B.65}$$

where the first identity uses the definition of $\boldsymbol{s}_t$, the second one the definition of convex conjugate and the last one is a consequence of the Fenchel-Young identity. We recall $\sigma_{\text{conv}(\mathcal{A})}$ is the support function of $\text{conv}(\mathcal{A})$.

□

Theorem 3.A. *Let f be convex, $\boldsymbol{x}_t$ denote the iterate generated by any of the proposed FW variants (AdaFW, AdaAFW, AdaPFW) after t iterations, with $N_t \geq 1$, and let $\boldsymbol{u}_t$ be defined recursively as $\boldsymbol{u}_0 = \nabla f(\boldsymbol{x}_0)$, $\boldsymbol{u}_{t+1} = (1-\xi_t)\boldsymbol{u}_t + \xi_t \nabla f(\boldsymbol{x}_t)$, where $\xi_t = 2/(\delta N_t + 2)$ if t is a good step and $\xi_t = 0$ otherwise. Then we have:*

$$h_t \leq f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t) \leq \frac{2\overline{L}_t \operatorname{diam}(\mathcal{A})^2}{\delta^2 N_t + \delta} + \frac{2(1-\delta)}{\delta^2 N_t^2 + \delta N_t}\big(f(\boldsymbol{x}_0) - \psi(\boldsymbol{u}_0)\big) = \mathcal{O}\left(\frac{1}{\delta^2 t}\right) . \quad \text{(B.66)}$$

Proof. The proof is structured as follows. First, we derive a bound for the case that k is a good step. Second, we derive a bound for the case that k is a bad step. Finally, we add over all iterates to derive the claimed bound.

In both the good and bad step cases, we'll make use of the auxiliary variable σ_k. This is defined recursively as $\sigma_0 = \psi(\nabla f(\boldsymbol{x}_k))$, $\sigma_{k+1} = (1 - \delta\xi_k)\sigma_k + \delta\xi_k \psi(\nabla f(\boldsymbol{x}_k))$. Since ψ is concave, Jensen's inequality gives that $\psi(\boldsymbol{u}_k) \geq \sigma_k$ for all k.

Case 1: k is a good step:

By Lemma 10, we have the following sequence of inequalities, valid for all $\xi_k \in [0, 1]$:

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) - \xi_k g_k + \frac{\xi_k^2 L_k}{2}\|\boldsymbol{d}_k\|^2 && \text{(B.67)} \\ &\leq f(\boldsymbol{x}_k) - \xi_k \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{s}_k\rangle + \frac{\xi_k^2 L_k}{2}\|\boldsymbol{d}_k\|^2 && \text{(B.68)} \\ &\leq (1-\delta\xi_k) f(\boldsymbol{x}_k) + \delta\xi_k \psi(\nabla f(\boldsymbol{x}_k)) + \frac{\xi_k^2 L_k}{2}\|\boldsymbol{d}_k\|^2 , && \text{(B.69)} \end{aligned}$$

where the second inequality follows from the definition of g_k (this is an equality for AdaFW but an inequality for the other variants) and the last inequality follows from Lemma 11.

Subtracting $\psi(\boldsymbol{u}_{k+1})$ from both sides of the previous inequality gives

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) - \psi(\boldsymbol{u}_{k+1}) &\leq f(\boldsymbol{x}_{k+1}) - \sigma_{k+1} && \text{(B.70)} \\ &\leq (1-\delta\xi_k)\big[f(\boldsymbol{x}_k) - \sigma_k\big] + \frac{\xi_k^2 L_k}{2}\|\boldsymbol{s}_k - \boldsymbol{x}_k\|^2 && \text{(B.71)} \end{aligned}$$

Let $\xi_k = 2/(\delta N_k + 2)$ and $a_k \stackrel{\text{def}}{=} \frac{1}{2}((N_k - 2)\delta + 2)((N_k - 1)\delta + 2)$. With these definitions, we have the following trivial identities that we will use soon:

$$\begin{aligned} a_{k+1}(1 - \delta\xi_k) &= \frac{1}{2}((N_k - 2)\delta + 2)((N_k - 1)\delta + 2) = a_k && \text{(B.72)} \\ a_{k+1}\frac{\xi_k^2}{2} &= \frac{((N_k - 1)\delta + 2)}{(N_k\delta + 2)} \leq 1 && \text{(B.73)} \end{aligned}$$

where in the first inequality we have used that k is a good step and so $N_{k+1} = N_k + 1$.

Multiplying Eqn. B.71 by a_{k+1} we have

$$a_{k+1}\big(f(\boldsymbol{x}_{k+1}) - \psi(\boldsymbol{u}_{k+1})\big) \leq a_{k+1}(1-\delta\xi_k)\big[f(\boldsymbol{x}_k) - \sigma_k\big] + \frac{L_k}{2}\|\boldsymbol{s}_k - \boldsymbol{x}_k\|^2 \tag{B.74}$$

$$\overset{Eqn.\ B.72}{=} a_k\big[f(\boldsymbol{x}_k) - \sigma_k\big] + \frac{L_k}{2}\|\boldsymbol{s}_k - \boldsymbol{x}_k\|^2 \tag{B.75}$$

$$\leq\ a_k\big[f(\boldsymbol{x}_k) - \sigma_k\big] + L_k \operatorname{diam}(\mathcal{A})^2 \tag{B.76}$$

Case 2: k is a bad step:
Lemma 6 with $\xi_k = 0$ guarantees that the objective function is non-increasing, i.e., $f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k)$. By construction of σ_k we have $\sigma_{k+1} = \sigma_k$, and so substracting both multiplied by a_{k+1} we obtain

$$a_{k+1}\big(f(\boldsymbol{x}_{k+1}) - \psi(\boldsymbol{u}_{k+1})\big) \leq a_{k+1}\big(f(\boldsymbol{x}_{k+1}) - \sigma_{k+1}\big) \tag{B.77}$$

$$\leq a_{k+1}\big(f(\boldsymbol{x}_k) - \sigma_k\big) \tag{B.78}$$

$$= a_k\big(f(\boldsymbol{x}_k) - \sigma_k\big)\ , \tag{B.79}$$

where in the last identity we have used that its a bad step and so $a_{k+1} = a_k$.
Final: combining cases and adding over iterates:
We can combine Eqn. B.76 and Eqn. B.79 into the following inequality:

$$a_{k+1}\big(f(\boldsymbol{x}_{k+1}) - \psi(\boldsymbol{u}_{k+1})\big) - a_k\big(f(\boldsymbol{x}_k) - \psi(\boldsymbol{u}_k)\big) \leq L_k \operatorname{diam}(\boldsymbol{A})^2 \mathbb{1}\{k \text{ is a good step}\}\ , \tag{B.80}$$

where $\mathbb{1}\{\text{condition}\}$ is 1 if condition is verified and 0 otherwise.

Adding this inequality from 0 to $t-1$ gives

$$a_t\big(f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t)\big) \leq \sum_{k=0}^{t-1} L_k Q_A^2 \mathbb{1}\{k \text{ is a good step}\} + a_0(f(\boldsymbol{x}_0) - \sigma_0) \tag{B.81}$$

$$= N_t \overline{L}_t \operatorname{diam}(\mathcal{A})^2 + (1-\delta)(2-\delta)(f(\boldsymbol{x}_0) - \sigma_0) \tag{B.82}$$

Finally, dividing both sides by a_t (note that $a_t > 0$ for $N_t \geq 1$) and using $(2-\delta) \leq 2$ we obtain

$$f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t) \leq \frac{2N_t}{((N_t-2)\delta+2)((N_t-1)\delta+2)} \overline{L}_t Q_{\mathcal{A}}^2 \tag{B.83}$$

$$+ \frac{4(1-\delta)}{((N_t-2)\delta+2)((N_t-1)\delta+2)} (f(\boldsymbol{x}_0) - \sigma_0) \tag{B.84}$$

We will now use the inequalities $(N_t-2)\delta + 2 \geq N_t\delta$ and $(N_t-1)\delta + 2 \geq N_t\delta + 1$ for the terms in the denominator to obtain

$$f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t) \leq \frac{2\overline{L}_t Q_{\mathcal{A}}^2}{\delta^2 N_t + \delta} + \frac{4(1-\delta)}{\delta_t^2 N_t^2 + \delta N_t} (f(\boldsymbol{x}_0) - f(\boldsymbol{x}^\star))\ . \tag{B.85}$$

which is the desired bound:

$$f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t) \leq \frac{2\overline{L}_t Q_{\mathcal{A}}^2}{\delta^2 N_t + \delta} + \frac{4(1-\delta)}{\delta_t^2 N_t^2 + \delta N_t}\big[f(\boldsymbol{x}_0) - \psi(\nabla f(\boldsymbol{x}_0))\big]. \tag{B.86}$$

We will now prove the bound $h_t \leq f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t)$. Let $\boldsymbol{u}^\star$ be an arbitrary maximizer of ψ. Then by duality we have that $f(\boldsymbol{x}^\star) = \psi(\boldsymbol{u}^\star)$ and so

$$f(\boldsymbol{x}_t) - \psi(\boldsymbol{u}_t) = f(\boldsymbol{x}_t) - f*(\boldsymbol{x}^\star) + \psi(\boldsymbol{u}^\star) - \psi(\boldsymbol{u}_t) \geq f(\boldsymbol{x}_t) - f*(\boldsymbol{x}^\star) = h_t \tag{B.87}$$

Finally, the $\mathcal{O}(\frac{1}{\delta t})$ rate comes from bounding the number of good steps from Eqn. 2.7, for which we have $1/N_t \leq \mathcal{O}(1/t)$, and bounding the Lipschitz estimate by a contant (Proposition 4). □

Matching Pursuit

Lemma 12. *Let $\boldsymbol{s}_t$ be as defined in AdaMP, $R_{\mathcal{B}}$ be the level set radius defined as*

$$R_{\mathcal{B}} = \max_{\substack{\boldsymbol{x}\in\text{lin}(\boldsymbol{A}) \\ f(\boldsymbol{x})\leq f(\boldsymbol{x}_0)}} \|\boldsymbol{x} - \boldsymbol{x}^\star\|_{\mathcal{B}} \ , \tag{B.88}$$

and $\boldsymbol{x}^\star$ be any solution to Eqn. B.1. Then we have

$$\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t\rangle \geq \frac{\delta}{\max\{\mathcal{R}_{\mathcal{B}}, 1\}}\big(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)\big) \tag{B.89}$$

Proof. By definition of atomic norm we have

$$\frac{\boldsymbol{x}_t - \boldsymbol{x}_t^\star}{\|\boldsymbol{x}_t - \boldsymbol{x}^\star\|_{\mathcal{B}}} \in \text{conv}(\boldsymbol{\mathcal{B}}) \tag{B.90}$$

Since $f(\boldsymbol{x}_t) \leq f(\boldsymbol{x}_0)$, which is a consequence of sufficient decrease condition (Eq. Eqn. B.52), we have that $R_{\mathcal{B}} \geq \|\boldsymbol{x}_t - \boldsymbol{x}^\star\|_{\mathcal{B}}$ and so $\zeta \stackrel{\text{def}}{=} \|\boldsymbol{x}_t - \boldsymbol{x}^\star\|_{\mathcal{B}}/R_{\mathcal{B}} \leq 1$. By symmetry of $\boldsymbol{\mathcal{B}}$ we have that

$$\frac{\boldsymbol{x}_t - \boldsymbol{x}^\star}{R_{\mathcal{B}}} = \zeta\frac{\boldsymbol{x}_t - \boldsymbol{x}^\star}{\|\boldsymbol{x}_t - \boldsymbol{x}^\star\|_{\mathcal{B}}} + (1-\zeta)\mathbf{0} \in \text{conv}(\boldsymbol{\mathcal{B}}) \ . \tag{B.91}$$

We will now use this fact to bound the original expression. By definition of $\boldsymbol{s}_t$ we have

$$\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t\rangle \overset{Eqn.\ B.2}{\geq} \delta \max_{\boldsymbol{s}\in\boldsymbol{\mathcal{B}}}\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}\rangle \tag{B.92}$$

$$\overset{Eqn.\ B.91}{\geq} \frac{\delta}{R_{\mathcal{B}}}\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x}^\star\rangle \tag{B.93}$$

$$\geq \frac{\delta}{R_{\mathcal{B}}}(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)) \tag{B.94}$$

where the last inequality follows by convexity.

□

Theorem 3.B. *Let f be convex, $\boldsymbol{x}^\star$ be an arbitrary solution to Eqn. B.1 and let $R_{\mathcal{B}}$ the level set radius:*

$$R_{\mathcal{B}} = \max_{\substack{\boldsymbol{x}\in\operatorname{lin}(\boldsymbol{A}) \\ f(\boldsymbol{x})\leq f(\boldsymbol{x}_0)}} \|\boldsymbol{x}-\boldsymbol{x}^\star\|_{\mathcal{B}} \ . \tag{B.95}$$

If we denote by $\boldsymbol{x}_t$ the iterate generated by AdaMP after $t \geq 1$ iterations and $\beta = \delta/R_{\mathcal{B}}$, then we have:

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \leq \frac{2\overline{L}_t \operatorname{radius}(\boldsymbol{A})^2}{\beta^2 t + \beta} + \frac{2(1-\beta)}{\beta^2 t^2 + \beta t} h_0 = \mathcal{O}\left(\frac{1}{\beta^2 t}\right) \ . \tag{B.96}$$

Proof. Let $\boldsymbol{x}^\star$ be an arbitrary solution to Eqn. B.1. Then by Lemma 6, we have the following sequence of inequalities, valid for all $\xi_t \geq 0$:

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \xi_k\langle -\nabla f(\boldsymbol{x}_k), \boldsymbol{s}_k\rangle + \frac{\xi_k^2 L_k}{2}\|\boldsymbol{s}_k\|^2 \tag{B.97}$$

$$\leq f(\boldsymbol{x}_k) - \xi_k \frac{\delta}{R_{\mathcal{B}}}\big[f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\big] + \frac{\xi_k^2 L_t}{2}\|\boldsymbol{s}_k\|^2 \ , \tag{B.98}$$

where the second inequality follows from Lemma 12.

Subtracting $f(\boldsymbol{x}^\star)$ from both sides of the previous inequality gives

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}^\star) \leq \left(1 - \frac{\delta}{R_{\mathcal{B}}}\xi_k\right)\big[f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\big] + \frac{\xi_k^2 L_k}{2}\|\boldsymbol{s}_k\|^2 \ . \tag{B.99}$$

Let $\beta = \delta/R_{\mathcal{B}}$ and $\xi_k = 2/(\beta k + 2)$ and $a_k \overset{\text{def}}{=} \frac{1}{2}((k-2)\beta+2)((k-1)\beta+2)$. With these definitions, we have the following trivial results:

$$a_{k+1}(1-\beta\xi_k) = \frac{1}{2}((k-2)\beta+2)((k-1)\beta+2) = a_k \tag{B.100}$$

$$a_{k+1}\frac{\xi_k^2}{2} = \frac{((k-1)\beta+2)}{(k\beta+2)} \leq 1 \ . \tag{B.101}$$

Multiplying Eqn. B.99 by a_{k+1} we have

$$a_{k+1}\big(f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}^\star)\big) \leq a_{k+1}(1-\beta\xi_k)\big[f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\big] + \frac{L_k}{2}\|\boldsymbol{s}_k\|^2 \tag{B.102}$$

$$\overset{Eqn.\ B.72}{=} a_k\big[f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\big] + \frac{L_k}{2}\|\boldsymbol{s}_k\|^2 \tag{B.103}$$

$$\leq \ a_k\big[f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\big] + L_t \operatorname{radius}(\mathcal{A})^2 \tag{B.104}$$

Adding this last inequality from 0 to $t-1$ gives

$$a_t\big(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)\big) \leq \sum_{k=0}^{t-1} L_k \operatorname{radius}(\boldsymbol{A})^2 + a_0(f(\boldsymbol{x}_0) - \beta_0) \tag{B.105}$$

$$= t\overline{L}_t \operatorname{diam}(\mathcal{A})^2 + (1-\delta)(2-\delta)(f(\boldsymbol{x}_0) - \beta_0) \tag{B.106}$$

Finally, dividing both sides by a_t (note that $a_1 = 2 - \beta \geq 1$ and so a_t is strictly positive for $t \geq 1$), and using $(2 - \delta) \leq 2$ we obtain

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \leq \frac{2t}{((t-2)\beta + 2)((t-1)\beta + 2)} \overline{L}_t \operatorname{radius}(\boldsymbol{A})^2 \tag{B.107}$$

$$+ \frac{4(1-\beta)}{((t-2)\beta + 2)((t-1)\beta + 2)} (f(\boldsymbol{x}_0) - \beta_0) \tag{B.108}$$

We will now use the inequalities $(t-2)\beta + 2 \geq t\beta$ and $(t-1)\beta + 2 \geq t\beta + 1$ to simplify the terms in the denominator. With this we obtain to obtain

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \leq \frac{2\overline{L}_t \operatorname{radius}(\boldsymbol{A})^2}{\beta^2 N_t + \beta} + \frac{4(1-\beta)}{\beta_t^2 N_t^2 + \beta N_t} (f(\boldsymbol{x}_0) - f(\boldsymbol{x}^\star)) \ , \tag{B.109}$$

which is the desired bound. □

B.6 Proofs of convergence for strongly convex objectives

The following proofs depend on some definitions of geometric constants, which are defined in B.2 as well as two crucial lemmas from [18, Appendix C].

Frank-Wolfe variants

We are now ready to present the convergence rate of the backtracking Frank–Wolfe variants. As we did in B.4, although the original proof combines the rates for FW variants and MP, the proof will be split into two, in which we prove separately the linear convergence rates for AdaAFW and AdaPFW (Theorem 4.A) and AdaMP (Theorem 4.B).

Theorem 4.A. *Let f be μ–strongly convex. Then for each good step we have the following geometric decrease:*

$$h_{t+1} \leq (1 - \rho_t) h_t, \tag{B.110}$$

with

$$\rho_t = \frac{\mu \delta^2}{4 L_t} \left(\frac{\mathrm{PWidth}(\boldsymbol{A})}{\mathrm{diam}(\mathrm{conv}(\mathcal{A}))} \right)^2 \qquad \textit{for AdaAFW} \tag{B.111}$$

$$\rho_t = \min \left\{ \frac{\delta}{2}, \delta^2 \frac{\mu}{L_t} \left(\frac{\mathrm{PWidth}(\boldsymbol{A})}{\mathrm{diam}(\mathrm{conv}(\mathcal{A}))} \right)^2 \right\} \qquad \textit{for AdaPFW} \tag{B.112}$$

Note. In the main paper we provided the simplified bound $\rho_t = \frac{\mu}{4 L_t} \left(\frac{\mathrm{PWidth}(\boldsymbol{A})}{\mathrm{diam}(\boldsymbol{A})} \right)^2$ for both algorithms AdaAFW and AdaPFW for simplicity. It is easy to see that the bound for AdaPFW above can be trivially bounded by this quantity by noting that $\delta^2 \leq \delta$ and that μ / L_t and $\mathrm{PWidth}(\boldsymbol{A}) / \mathrm{diam}(\mathrm{conv}(\mathcal{A}))$ are necessarily smaller than 1.

Proof. The structure of this proof is similar to that of [18, Theorem 8]. We begin by upper bounding the suboptimality h_t. Then we derive a lower bound on $h_{t+1} - h_t$. Combining both we arrive at the desired geometric decrease.

Upper bounding h_t
Assume $\boldsymbol{x}_t$ is not optimal, ie $h_t > 0$. Then we have $\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{x}^\star - \boldsymbol{x}_t \rangle > 0$. Using the definition of the geometric strong convexity bound and letting $\overline{\gamma} \stackrel{\text{def}}{=} \gamma(\boldsymbol{x}_t, \boldsymbol{x}^\star)$ we have

$$\begin{aligned}
\frac{\overline{\gamma}^2}{2} \mu_f^A &\leq f(\boldsymbol{x}^\star) - f(\boldsymbol{x}_t) + \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{x}^\star - \boldsymbol{x}_t \rangle && \text{(B.113)} \\
&= -h_t + \overline{\gamma} \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_f(\boldsymbol{x}_t) - \boldsymbol{v}_f(\boldsymbol{x}_t) \rangle && \text{(B.114)} \\
&\leq -h_t + \overline{\gamma} \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{v}_t \rangle && \text{(B.115)} \\
&= -h_t + \overline{\gamma} q_t \ , && \text{(B.116)}
\end{aligned}$$

where $q_t \stackrel{\text{def}}{=} \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{v}_t \rangle$. For the last inequality we have used the definition of $\boldsymbol{v}_f(\boldsymbol{x})$ which implies $\langle f(\boldsymbol{x}_t), \boldsymbol{v}_f(\boldsymbol{x}_t) \rangle \leq \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{v}_t \rangle$ and the fact that $\boldsymbol{s}_t = \boldsymbol{s}_f(\boldsymbol{x}_t)$. Therefore

$$h_t \leq -\frac{\overline{\gamma}^2}{2}\mu_f^A + \overline{\gamma} q_t \ , \tag{B.117}$$

which can always be upper bounded by taking $\overline{\gamma} = \mu^{-1} q_t$ (since this value of $\overline{\gamma}$ maximizes the expression on the right hand side of the previous inequality) to arrive at

$$h_t \leq \frac{q_t^2}{2\mu_f^A} \tag{B.118}$$

$$\leq \frac{q_t^2}{2\mu\Delta^2} \ , \tag{B.119}$$

with $\Delta \stackrel{\text{def}}{=} \text{PWidth}(\boldsymbol{A})$ and where the last inequality follows from Lemma 5.

Lower bounding progress $h_t - h_{t+1}$.
Let G be defined as $G = 1/2$ for AdaAFW and $G = 1$ for AdaPFW. We will now prove that for both algorithms we have

$$\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle \geq \delta G q_t \ . \tag{B.120}$$

For AdaAFW, by the way the direction $\boldsymbol{d}_t$ is chosen on Line 6, we have the following sequence of inequalities:

$$\begin{aligned} 2\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle &\geq \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t^{FW} \rangle + \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t^{A} \rangle \\ &\geq \delta \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle + \delta \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{v}_t \rangle \\ &= \delta \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{v}_t \rangle \\ &= \delta q_t \ , \end{aligned}$$

For AdaPFW, since $\boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{v}_t$, it follows from the definition of q_t that $\langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle \geq \delta q_t$.

We split the rest of the analysis into three cases: $\gamma_t < \gamma_t^{\max}, \gamma_t = \gamma_t^{\max} \geq 1$ and $\gamma_t = \gamma_t^{\max} < 1$. We prove a geometric descent in the first two cases. In the case where $\gamma_t = \gamma_t^{\max} < 1$ (a bad step) we show that the number of bad steps is bounded.

Case 1: $\gamma_t < \gamma_t^{\max}$:
By Lemma 6, we have

$$f(\boldsymbol{x}_{t+1}) = f(\boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t) \leq f(\boldsymbol{x}_t) + \min_{\eta \in [0, \gamma_t^{\max}]} \left\{ \eta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \eta^2}{2} \|\boldsymbol{d}_t\|^2 \right\} \tag{B.121}$$

Because $\gamma_t < \gamma_t^{\max}$ and since the expression inside the minimization term Eqn. B.121 is a convex function of η, the minimizer is unique and it coincides with the minimum of the unconstrained problem. Hence we have

$$\min_{\eta \in [0, \gamma_t^{\max}]} \left\{ \eta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \eta^2}{2} \|\boldsymbol{d}_t\|^2 \right\} = \min_{\eta \geq 0} \left\{ \eta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \eta^2}{2} \|\boldsymbol{d}_t\|^2 \right\} \tag{B.122}$$

Replacing in Eqn. 6, our bound becomes

$$f(\boldsymbol{x}_{t+1}) = f(\boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t) \leq f(\boldsymbol{x}_t) + \min_{\eta \geq 0} \left\{ \eta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \eta^2}{2} \|\boldsymbol{d}_t\|^2 \right\} \tag{B.123}$$

$$\leq f(\boldsymbol{x}_t) + \min_{\eta \geq 0} \left\{ \eta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \eta^2}{2} M^2 \right\} \tag{B.124}$$

$$\leq f(\boldsymbol{x}_t) + \eta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle + \frac{L_t \eta^2}{2} M^2, \;\; \forall \eta \geq 0 \tag{B.125}$$

where the second inequality comes from bounding $\|\boldsymbol{d}_t\|$ by $M \stackrel{\text{def}}{=} \text{diam}(\text{conv}(\mathcal{A}))$. Subtracting $f(\boldsymbol{x}^\star)$ from both sides and rearranging we have

$$h_t - h_{t+1} \geq \eta \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t \rangle - \frac{1}{2} \eta^2 L_t M^2, \;\; \forall \eta \geq 0 \; . \tag{B.126}$$

Using the gap inequality Eqn. B.120 our lower bound becomes

$$h_t - h_{t+1} \geq \eta \delta G q_t - \frac{1}{2} \eta^2 L_t M^2, \;\; \forall \eta \geq 0 \; . \tag{B.127}$$

Noting that the lower bound in Eqn. B.127 is a concave function of η, we maximize the bound by selecting $\eta^\star = (L_t M^2)^{-1} \delta G q_t$. Plugging $\eta^\star$ into the bound in Eqn. B.127 and then using the strong convexity bound Eqn. B.119 we have

$$h_t - h_{t+1} \geq \frac{\mu G^2 \Delta^2 \delta^2}{L_t M^2} h_t \Longrightarrow h_{t+1} \leq \Big(1 - \frac{\mu G^2 \Delta^2 \delta^2}{L_t M^2}\Big) h_t \; . \tag{B.128}$$

Then we have geometric convergence with rate $1 - \rho$ where $\rho = (4 L_t M^2)^{-1} \mu \Delta^2 \delta^2$ for AdaAFW and $\rho = (L_t M^2)^{-1} \mu \Delta^2 \delta^2$ for AdaPFW.

Case 2: $\gamma_t = \gamma_t^{\max} \geq 1$

By Lemma 6 and the gap inequality Eqn. B.120, we have

$$h_t - h_{t+1} = f(\boldsymbol{x}_t) - f(\boldsymbol{x}_{t+1}) \geq \eta \delta G q_t - \frac{1}{2} \eta^2 L_t M^2, \;\; \forall \eta \leq \gamma_t^{\max} \; . \tag{B.129}$$

Since the lower bound Eqn. B.129 is true for all $\eta \leq \gamma_t^{\max}$, we can maximize the bound with $\eta^\star = \min\{(L_t M^2)^{-1} \delta G q_t, \gamma_t^{\max}\}$. In the case when $\eta^\star = (L_t M^2)^{-1} \delta G q_t$ we get the same bound as we do in Eqn. B.128 and hence have linear convergence with rate $1 - \rho$ where $\rho = (4 L_t M^2)^{-1} \mu \Delta^2 \delta^2$ for AdaAFW and $\rho = (L_t M^2)^{-1} \mu \Delta^2 \delta^2$ for AdaPFW. If $\eta^\star = \gamma_t^{\max}$ then this implies $L_t M^2 \leq \delta G q_t$. Since $\gamma_t^{\max}$ is assumed to be greater than 1 and the bound holds for all $\eta \leq \gamma_t^{\max}$ we have in particular that it holds for $\eta = 1$ and hence

$$h_t - h_{t+1} \geq \delta G q_t - \frac{1}{2} L_t M^2 \tag{B.130}$$

$$\geq \delta G q_t - \frac{\delta G q_t}{2} \tag{B.131}$$

$$\geq \frac{\delta G h_t}{2} \; , \tag{B.132}$$

where in the second line we use the inequality $L_t M^2 \leq \delta G q_t$ and in the third we use the inequality $h_t \leq q_t$ which is an immediate consequence of convexity of f. Then we have

$$h_{t+1} \leq (1-\rho) h_t \ , \tag{B.133}$$

where $\rho = \delta/4$ for AdaAFW and $\rho = \delta/2$ for AdaPFW. Note by Proposition 3 and the fact $\mu \leq L_t$ we have $\delta/4 \geq (4L_t M^2)^{-1} \mu \Delta^2 \delta^2$.

Case 3: $\gamma_t = \gamma_t^{\max} < 1$ **(bad step)**
In this case, we have either a drop or swap step and can make no guarantee on the progress of the algorithm (drop and swap are defined in B.3). For AdaAFW, $\gamma_t = \gamma_t^{\max} < 1$ is a drop step. From lines 6–10 of AdaAFW we can make the following distinction of cases. In case of a FW step, then $\mathcal{S}_{t+1} = \{\boldsymbol{s}_t\}$ and $\gamma_t = \gamma_t^{\max} = 1$, otherwise $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{\boldsymbol{s}_t\}$. In case of an Away step, $\mathcal{S}_{t+1} = \mathcal{S}_t \backslash \{\boldsymbol{v}_t\}$ if $\gamma_t = \gamma_t^{\max} < 1$, otherwise $\mathcal{S}_{t+1} = \mathcal{S}_t$. Note a drop step can only occur at an Away step. For AdaPFW, $\gamma_t = \gamma_t^{\max} < 1$ will be a drop step when $\boldsymbol{s}_t \in \mathcal{S}_t$ and will be a swap step when $\boldsymbol{s}_t \notin \mathcal{S}_t$.

Even though at these bad steps we do not have the same geometric decrease, Lemma 8 yields that the sequence $\{h_t\}$ is a non-increasing sequence, i.e., $h_{t+1} \leq h_t$. Since we are guaranteed a geometric decrease on steps that are not bad steps, the bounds on the number of bad steps of Eq. Eqn. 2.7 is sufficient to conclude that AdaAFW and AdaPFW exhibit a global linear convergence.

□

Matching Pursuit

We start by proving the following lemma, which will be crucial in the proof of the backtracking MP's linear convergence rate.

Lemma 13. *Suppose that $\boldsymbol{A}$ is a non-empty compact set and that f is μ–strongly convex. Let $\nabla_{\mathcal{B}} f(\boldsymbol{x})$ denote the orthogonal projection of $\nabla f(\boldsymbol{x})$ onto $\operatorname{lin}(\mathcal{B})$. Then for all $\boldsymbol{x}^\star - \boldsymbol{x} \in \operatorname{lin}(\boldsymbol{A})$, we have*

$$f(\boldsymbol{x}^\star) \geq f(\boldsymbol{x}) - \frac{1}{2\mu \operatorname{mDW}(\mathcal{B})^2} \|\nabla_{\mathcal{B}} f(\boldsymbol{x})\|_{\mathcal{B}^\star}^2 \ . \tag{B.134}$$

Proof. From Locatello et al. [26, Theorem 6], we have that if f is μ-strongly convex, then

$$\mu_{\mathcal{B}} \stackrel{\text{def}}{=} \inf_{\boldsymbol{x},\boldsymbol{y} \in \operatorname{lin}(\mathcal{B}), \boldsymbol{x} \neq \boldsymbol{y}} \frac{2}{\|\boldsymbol{y}-\boldsymbol{x}\|_{\mathcal{B}}^2} [f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle] \tag{B.135}$$

is positive and verifies $\mu_{\mathcal{B}} \geq \operatorname{mDW}(\mathcal{B})^2 \mu$. Replacing $\boldsymbol{y} = \boldsymbol{x} + \gamma(\boldsymbol{x}^\star - \boldsymbol{x})$ in the definition above we have

$$f(\boldsymbol{x} + \gamma(\boldsymbol{x}^\star - \boldsymbol{x})) \geq f(\boldsymbol{x}) + \gamma \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x} \rangle + \gamma^2 \frac{\mu_{\mathcal{B}}}{2} \|\boldsymbol{x}^\star - \boldsymbol{x}\|_{\mathcal{B}}^2 \ . \tag{B.136}$$

We can fix $\gamma = 1$ on the left hand side and since the expression on the right hand side is true for all γ, we minimize over γ to find $\gamma^* = -\langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x}\rangle / \mu_{\mathcal{B}} \|\boldsymbol{x}^\star - \boldsymbol{x}\|_{\mathcal{B}}^2$. Thus the lower bound becomes

$$f(\boldsymbol{x}^\star) \geq f(\boldsymbol{x}) - \frac{1}{2\mu_{\mathcal{B}}} \frac{\langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x}\rangle}{\|\boldsymbol{x}^\star - \boldsymbol{x}\|_{\mathcal{B}}^2} \tag{B.137}$$

$$\geq f(\boldsymbol{x}) - \frac{1}{2\mu\, \mathrm{mDW}(\mathcal{B})^2} \frac{\langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x}\rangle}{\|\boldsymbol{x}^\star - \boldsymbol{x}\|_{\mathcal{B}}^2} \tag{B.138}$$

$$= f(\boldsymbol{x}) - \frac{1}{2\mu\, \mathrm{mDW}(\mathcal{B})^2} \frac{\langle \nabla_{\mathcal{B}} f(\boldsymbol{x}), \boldsymbol{x}^\star - \boldsymbol{x}\rangle}{\|\boldsymbol{x}^\star - \boldsymbol{x}\|_{\mathcal{B}}^2} \tag{B.139}$$

$$\geq f(\boldsymbol{x}) - \frac{1}{2\mu\, \mathrm{mDW}(\mathcal{B})^2} \|\nabla_{\mathcal{B}} f(\boldsymbol{x})\|_{\mathcal{B}^*}^2 \ , \tag{B.140}$$

where the last inequality follows by $|\langle \boldsymbol{y}, \boldsymbol{z}\rangle| \leq \|\boldsymbol{y}\|_{\mathcal{B}^*} \|\boldsymbol{z}\|_{\mathcal{B}}$ □

Theorem 4.B. *(Convergence rate backtracking MP) Let f be μ–strongly convex and suppose $\mathcal{B}$ is a non-empty compact set. Then AdaMP verifies the following geometric decrease for each $t \geq 0$:*

$$h_{t+1} \leq \Big(1 - \delta^2 \rho_t\Big) h_t, \quad \textit{with } \rho_t = \frac{\mu}{L_t} \left(\frac{\mathrm{mDW}(\mathcal{B})}{\mathrm{radius}(\mathcal{B})}\right)^2 \ , \tag{B.141}$$

where $\mathrm{mDW}(\mathcal{B})$ *the minimal directional width of* $\mathcal{B}$.

Proof. By Lemma 6 and bounding $\|\boldsymbol{d}_t\|$ by $R = \mathrm{radius}(\mathcal{B})$ we have

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) + \min_{\eta \in \mathbb{R}} \left\{ \eta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t\rangle + \frac{\eta^2 L_t R^2}{2} \right\} \tag{B.142}$$

$$= f(\boldsymbol{x}_t) - \frac{\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t\rangle^2}{2L_t R^2} \tag{B.143}$$

$$\leq f(\boldsymbol{x}_t) - \delta^2 \frac{\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t^\star\rangle^2}{2L_t R^2} \tag{B.144}$$

where $\boldsymbol{s}_t^\star$ is any element such that $\boldsymbol{s}_t^\star \in \arg\min_{\boldsymbol{s} \in \mathcal{B}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}\rangle$ and the inequality follows from the optimality of min and the fact that $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t^\star\rangle \leq 0$. Let $\nabla_{\mathcal{B}} f(\boldsymbol{x}_t)$ denote as in Lemma 13 the orthogonal projection of $\nabla f(\boldsymbol{x}_t)$ onto $\mathrm{lin}(\mathcal{B})$. Then the previous inequality simplifies to

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \delta^2 \frac{\langle \nabla_{\mathcal{B}} f(\boldsymbol{x}_t), \boldsymbol{s}_t^\star\rangle^2}{2L_t R^2} \ . \tag{B.145}$$

By definition of dual norm, we also have $\langle -\nabla_{\mathcal{B}} f(\boldsymbol{x}_t), \boldsymbol{s}_t^\star \rangle = \|\nabla_{\mathcal{B}} f(\boldsymbol{x}_t)\|_{\mathcal{B}*}^2$. Subtracting $f(\boldsymbol{x}^\star)$ from both sides we obtain the upper-bound:

$$h_{t+1} \leq h_t - \delta^2 \frac{\|\nabla_{\mathcal{B}} f(\boldsymbol{x}_t)\|_{\mathcal{B}*}^2}{2L_t R^2} \tag{B.146}$$

To derive the lower-bound, we use Lemma 13 with $\boldsymbol{x} = \boldsymbol{x}_t$ and see that

$$\|\nabla_{\mathcal{B}} f(\boldsymbol{x}_t)\|_{\mathcal{B}^*} \geq 2\mu \, \mathrm{mDW}(\mathcal{B})^2 h_t \tag{B.147}$$

Combining the upper and lower bound together we have

$$h_{t+1} \leq \Big(1 - \delta^2 \frac{\mu \, \mathrm{mDW}(\mathcal{B})^2}{L_t R^2}\Big) h_t \ , \tag{B.148}$$

which is the claimed bound. □

B.7 Experiments

In this appendix we give give some details on the experiments which were omitted from the main text, as well as an extended set of results.

ℓ_1-regularized logistic regression, Madelon dataset

For the first experiment, we consider an ℓ_1-regularized logistic regression of the form

$$\arg\min_{\|\boldsymbol{x}\|_1 \leq \beta} \frac{1}{n} \sum_{i=1}^{n} \varphi(\boldsymbol{a}_i^\top \boldsymbol{x}, \boldsymbol{b}_i) + \frac{\lambda}{2} \|\boldsymbol{x}\|_2^2 \,, \tag{B.149}$$

where φ is the logistic loss. The linear subproblems in this case can be computed exactly ($\delta = 1$) and consists of finding the largest entry of the gradient. The regularization parameter λ is always set to $\lambda = \frac{1}{n}$.

We first consider the case in which the data $\boldsymbol{a}_i, \boldsymbol{b}_i$ is the Madelon datset. Below are the curves objective suboptimality vs time for the different methods considered. The regularization parmeter, denoted ℓ_1 ball radius in the figure, is chosen as to give 1%, 5% and 20% of non-zero coefficients (the middle figure is absent from the main text).

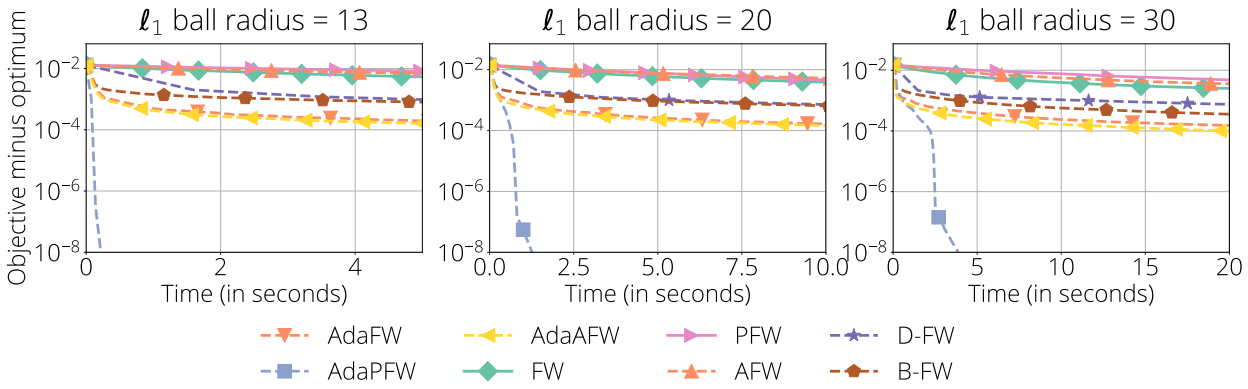


Figure B.1: **Comparison of different FW variants**. Problem is ℓ_1-regularized logistic regression and dataset is Madelon in the first, RCV1 in the second figure.

ℓ_1-regularized logistic regression, RCV1 dataset

The second experiment is identical to the first one, except the madelon datset is replaced by the larger RCV1 datset. Below we display the results of the comparison in this dataset:

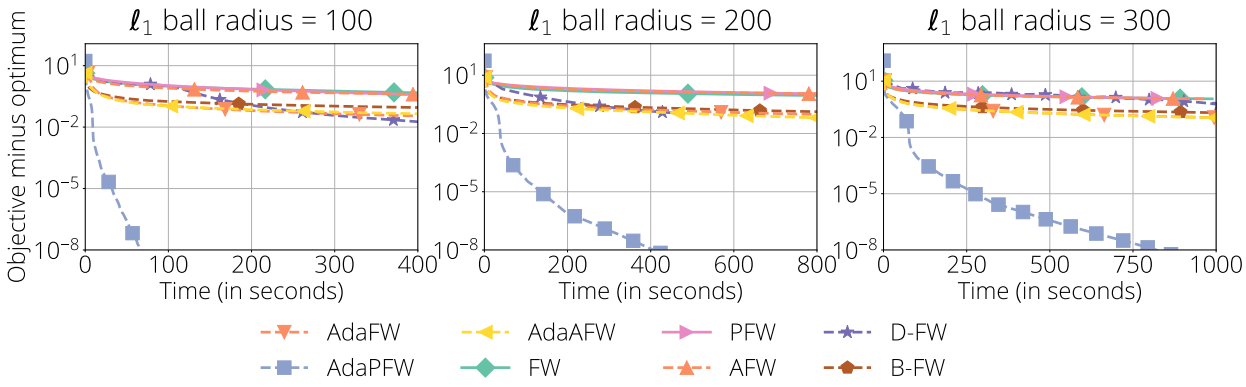


Figure B.2: **Comparison of different FW variants**. Problem is ℓ_1-regularized logistic regression and dataset is RCV1.

Nuclear norm-regularized Huber regression, MovieLens dataset

For the third experiment, we consider a collaborative filtering problem with the Movielens 1M dataset [33] as provided by the spotlight[1] Python package.

In this case the dataset consists of a sparse matrix $\boldsymbol{A}$ representing the ratings for the different movies and users. We denote by $\mathcal{I}$ the non-zero indices of this matrix. Then the optimization probllem that we consider is the following

$$\arg\min_{\|\boldsymbol{X}\|_* \leq \beta} \frac{1}{n} \sum_{(i,j)\in\mathcal{I}}^{n} L_\xi(\boldsymbol{A}_{i,j} - \boldsymbol{X}_{i,j}) \ , \tag{B.150}$$

where L_ξ is the Huber loss, defined as

$$L_\xi(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \xi, \\ \xi(|a| - \frac{1}{2}\xi), & \text{otherwise}. \end{cases} \tag{B.151}$$

The Huber loss is a quadratic for $|a| \leq \xi$ and grows linearly for $|a| > \xi$. The parameter ξ controls this tradeoff and was set to 1 during the experiments.

We compared the variant of FW that do not require to store the active set on this problem (as these are the only competitive variants for this problem).

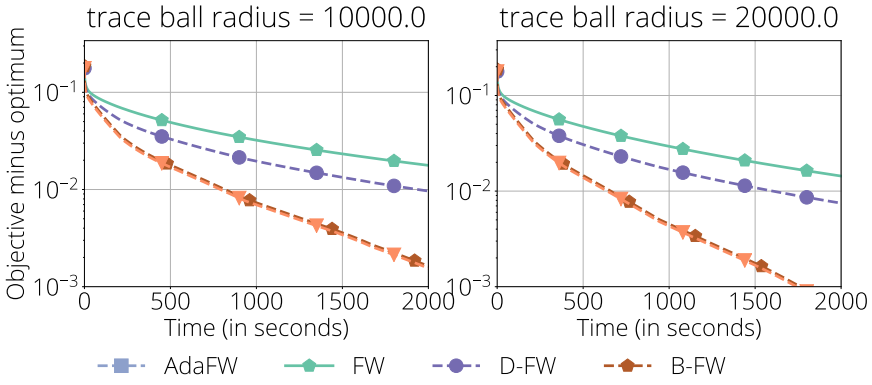


Figure B.3: **Comparison of different FW variants**. Comparison of FW variants on the Movielens 1M dataset.

[1] `https://github.com/maciejkula/spotlight`

Appendix C

Learning differentiable solvers for systems with hard constraints

C.1 1D convection

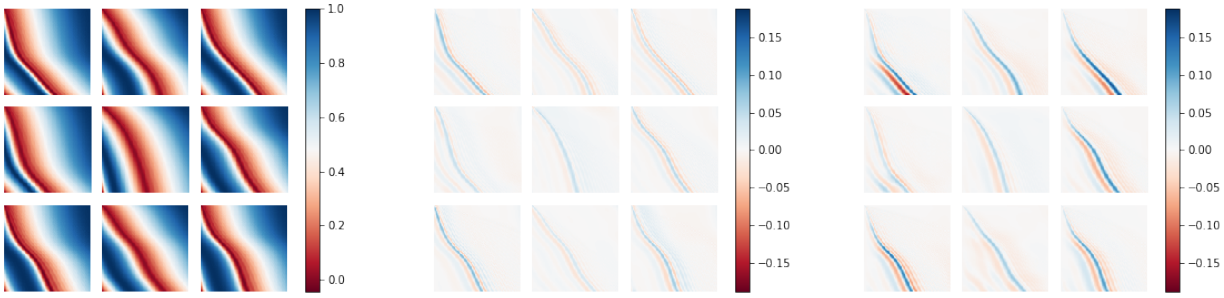
We study a 1D convection problem, describing transport phenomena. The problem can be formulated as follows:

$$\begin{aligned} &\frac{\partial u(x,t)}{\partial t} + \beta(x)\frac{\partial u(x,t)}{\partial x} = 0, && x \in (0,1), t \in (0,1), \\ &h(x) = \sin(\pi x), && x \in (0,1) \\ &g(t) \ = \sin\left(\frac{\pi}{2}t\right), && t \in (0,1). \end{aligned} \tag{C.1}$$

Here, $h(x)$ is the initial condition (at $t = 0$), $g(t)$ is the boundary condition (at $x = 0$), and $\beta(x)$ represents the variable coefficients (denoted by ϕ in Section 5.3). Given a set of variable coefficients, $\beta(x)$, and spatiotemporal points (x_i, t_i), the goal is to predict the correct solution $u(x,t)$. The $\beta(x)$ values are generated in the same manner as in Wang, Wang, and Perdikaris [64] via $\beta(x) = v(x) - \min_x v(x) + 1$, where $v(x)$ is generated from a Gaussian random field with a length scale of 0.2. We use a physics-informed DeepONet baseline model [64], trained with the PDE residual loss. Our hard-constrained model is composed of stacked dense layers and our PDE-CL, which allows for a fair comparison. We provide more details on our setup and experiments in Appendix C.2.

Results. We plot example heatmaps from the test set in Figure C.1.1. We compare how close our hard-constrained model is to the target solution (Figure C.1.1 b)), and similarly for the soft-constrained baseline model (Figure C.1.1 c)). The solution found by our hard-constrained model is much closer to the target solution than the solution found by the baseline model, and our model captures "sharp" features in the solution visibly better than the baseline model.

During the training procedure for both hard- and soft-constrained models, we track error on an unseen test set of PDE solutions with different PDE parameters from the training



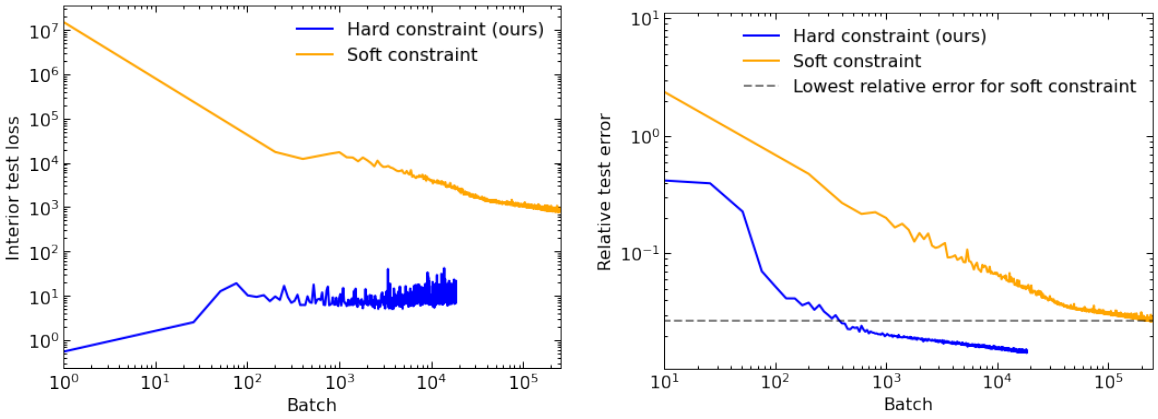
(a) Target (b) Hard-constrained difference (c) Soft-constrained difference

Figure C.1.1: **Heatmaps of 1D convection example test set predictions.** We compare our hard-constrained model and the baseline soft-constrained model on a test set of new wavespeed parameters β. Both architectures are the same, except for our additional PDE-CL in the hard-constrained model. a) Target solutions of a subset of PDEs in the test set. b) Difference between the predictions of our hard-constrained model and the target solution. c) Difference between the predictions of the baseline soft-constrained model and the target solution. Over the test dataset, our model achieves $\mathbf{1.32\% \pm 0.02\%}$ relative error and $\mathbf{9.84 \pm 2.15}$ PDE residual test loss. In contrast, the soft-constrained model only reaches $\mathbf{2.59\% \pm 0.15\%}$ relative error and $\mathbf{774 \pm 1.2}$ PDE residual test loss. Our model achieves $\mathbf{49\%}$ less relative error than the soft-constrained model. The errors in both models are concentrated around the "sharp" features in the solution, but these errors have higher magnitude in the soft-constrained model.

set. We show these error plots in Figure C.1.2. In Figure C.1.2 a), the PDE residual loss for the hard-constrained model starts close to six orders of magnitude lower than for the soft-constrained model, and it continues to remain low. In Figure C.1.2 b), we track the relative error with respect to the target solution obtained via a Lax-Wendroff scheme. Similarly, we see that our model starts at much smaller relative error immediately and also continues to decrease. Our model achieves $\mathbf{1.32\% \pm 0.02\%}$ relative error and $\mathbf{9.84 \pm 2.15}$ PDE residual test loss, versus $\mathbf{2.59\% \pm 0.15\%}$ and $\mathbf{774 \pm 1.2}$ for the soft-constrained baseline model. On the relative error metric, our model achieves a $\mathbf{49\%}$ **improvement** over the soft-constrained model. The standard deviation of our model for the relative error metric over the test dataset is also small. Our model trains faster than the soft-constrained method, due to much higher gains in accuracy per batch, even though each batch is slower (see Figure C.2.1).

C.2 Details on the 1d Convection problem

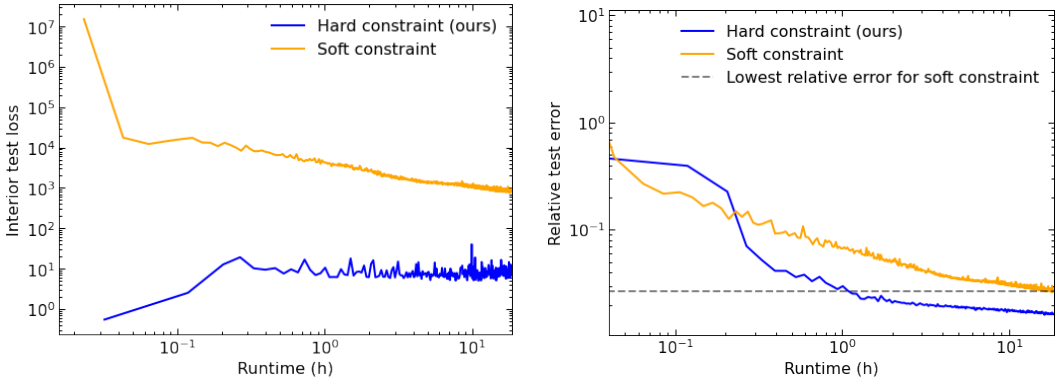
Experiment setup and implementation details. In this setting, the inputs to the models are two sets of $((x_1, t_1), \ldots, (x_n, t_n))$, and $((x'_1, t'_1), \ldots, (x'_{n'}, t'_{n'}))$ sampled points within the domain $\mathcal{X}$ (interior points) and the corresponding $\beta(x)$ values. We use the former for



(a) PDE residual loss on test set

(b) Relative error on test set

Figure C.1.2: **1D convection: Error on test set during training.** We train a NN with the PDE residual loss function ("soft constraint" baseline) and the same NN architecture with our PDE-CL ("hard constraint"). During training, we track error on the test set, which we plot on a log-log scale. a) PDE residual loss on the test set, during training. We observe that the NN starts by fitting the initial and boundary condition regression loss during training, which explains why the PDE residual loss seems to go up initially. b) Relative error on the test set, during training. Both measures show that our hard-constrained model starts at a much lower error on the test set at the very start of training. The grey, dashed line shows that the hard-constrained model achieves the same relative error as the soft-constrained model in over 100x fewer iterations, and ultimately achieves lower relative error. Wall-time comparison figures are given in Appendix C.2.



(a) PDE residual loss on test set

(b) Relative error on test set

Figure C.2.1: **Walltime plots for 1D convection**. During the training procedure, we track error on an unseen test set. Our hard- constrained model reaches the optimal accuracy of the soft-constrained model in 10x less time.

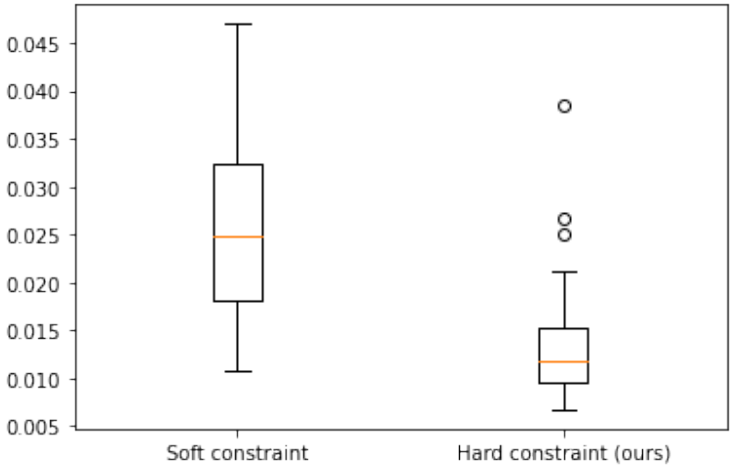


Figure C.2.2: **1D convection: Box plots showing error over test set**. We show the distribution of errors over the test set, at the end of training. Our hard-constrained model has both a lower error and a lower standard deviation as compared to the soft-constrained model.

fitting the PDE-CL, and the latter for computing the residual loss function. We also require a set $((x_{n+1}, t_{n+1}), \ldots, (x_{n+n'}, t_{n+n'}))$ of sampled points on the initial condition ($t = 0$) and boundary condition ($x = 0$). The training optimization problem is formulated as follows:

$$\min_{\theta} \sum_{\beta} \mathcal{L}_\beta(u_\theta) + \|\mathcal{F}_\beta(u_\theta; x'_1, \ldots, x'_{n'})\|_2^2 \text{ s.t. } \forall \beta,\ \mathcal{F}_\beta(u_\theta; x_1, \ldots, x_n) = 0, \tag{C.2}$$

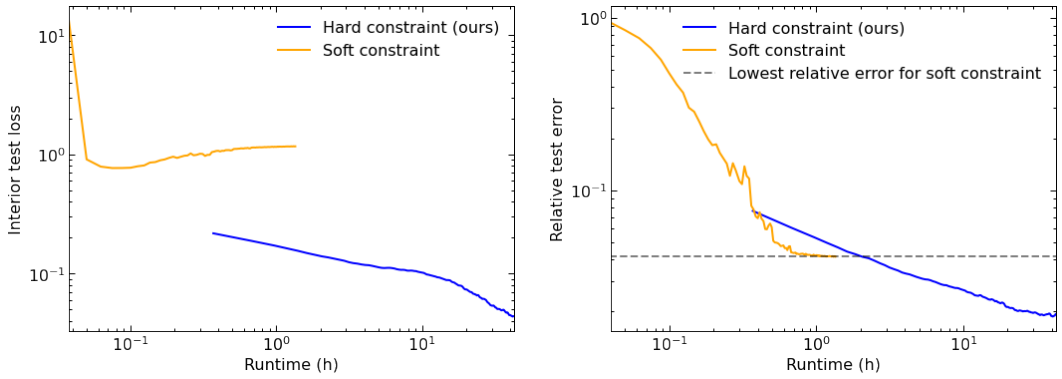
with,

$$\mathcal{L}_\beta(u_\theta) = \frac{1}{2} \sum_{i=1}^{n'} (u_\theta(\beta, x_{n+i}, t_{n+i}) - u(\beta, x_{n+i}, t_{n+i}))^2,$$

$$\mathcal{F}_\beta(u_\theta; x_1, \ldots, x_n) = \begin{pmatrix} \dfrac{\partial u_\theta(\beta, x_1, t_1)}{\partial t} + \beta(x_1) \dfrac{\partial u_\theta(\beta, x_1, t_1)}{\partial x} \\ \vdots \\ \dfrac{\partial u_\theta(\beta, x_n, t_n)}{\partial t} + \beta(x_n) \dfrac{\partial u_\theta(\beta, x_n, t_n)}{\partial x} \end{pmatrix},$$

where θ corresponds to parameters of the NN, $u(x, t)$ is the solution at the initial and boundary conditions, and $\mathcal{F}(u_\theta)$ is the PDE constraint that must be satisfied. The loss term $\mathcal{L}(u_\theta)$ is a regression loss over the initial and boundary conditions. The forward pass of the PDE-CL solves the following equality constrained problem,

$$\min_{\omega} \ \mathcal{L}(f_\theta^\top \omega) \quad \text{s.t.} \quad \mathcal{F}_\beta(f_\theta)^\top \omega = 0, \tag{C.3}$$



(a) PDE residual loss on test set

(b) Relative error on test set

Figure C.3.1: **Walltime plots for Darcy Flow**. During the training procedure, we track error on an unseen test set. Our hard- constrained model achieves higher accuracy much more quickly than the soft-constrained model.

where f_θ refers to the outputs of the base NN, on which we stack the PDE-CL. Since the initial/boundary condition regression loss uses a quadratic penalty, this equality constrained problem is in fact a convex equality constrained quadratic problem (EqQP), which is equivalent to a linear system. We solve this linear system using GMRES [94]. We compute the Jacobian via implicit differentiation with respect to Equation C.3.

In practice, we sample 750 points for the PDE-CL, and sample a separate 250 points for computing the residual in the loss function. To ensure fairness, we sample 1000 points for the soft-constrained method, which are all used to compute the residual in the loss function. We use N=600 for the number of basis functions in the PDE-CL.

We show plots against wall time in Figure C.2.1, and the distributions of errors over the test set in Figure C.2.2.

C.3 Details on the Darcy Flow problem

Experiment setup and implementation details. Our goal is to find parameters θ, which solve,

$$\begin{aligned} &\min_{\theta} \sum_{\nu} \|\mathcal{F}_\nu(u_\theta) - f(x)\|_2^2 \\ &\text{s.t. } \forall \nu,\ \mathcal{F}_\nu(u_\theta) = f(x). \end{aligned} \tag{C.4}$$

By design, the objective function for feasible parameters θ is zero. While numerical issues may prevent exact feasibility, solving the equality constrained problem by additionally minimizing the PDE residual helps the training procedure. The soft-constrained training method is trained only by minimizing the PDE residual. We train the FNO model using the same

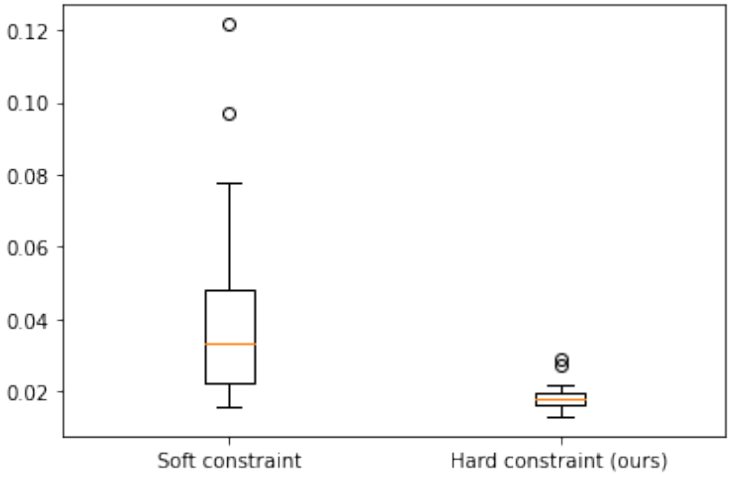


Figure C.3.2: **2D Darcy Flow: Box plots showing error over test set**. We show the distribution of errors over the test set, at the end of training. Our hard-constrained model has both a lower error, as well as a significantly lower standard deviation as compared to the soft-constrained model.

hyperparameters as Li et al. [62]. We denote the FNO model part of the architecture as f_θ. The PDE-CL constrains the output of the FNO model by solving the linear system,

$$\forall \nu,\ \mathcal{F}_\nu(f_\theta)^\top \omega = f(x). \tag{C.5}$$

To train our model, we compute the Jacobian of this layer via implicit differentiation, with respect to this linear system equation.

We sample 3721 points for the PDE-CL. We also sample 3721 points for the soft-constrained method, which are all used to compute the residual in the loss function. We use N=4000 for the number of basis functions in the PDE-CL.

We show plots against wall time in Figure C.3.1, and the distributions of errors over the test set in Figure C.3.2.

C.4 Hard constraints bound

To provide an evaluation of how hard the hard constraints are, we conduct an additional study where we look at the error in prediction for points sampled and used to fit the PDE-CL against points that were not sampled. With a trained hard-constrained model for 1D convection, we sample a batch of points and create a density plot showing the error as a function of points used for fitting the PDE-CL and points not used for fitting the PDE-CL. The histogram in Figure C.4.1 shows that the errors are qualitatively the same between points used for fitting the PDE-CL and those not used. There are 1000 points used for fitting the PDE-CL, and 9000 points not used. Our results show that our model achieves low error, even outside of the points used for fitting the PDE-CL.

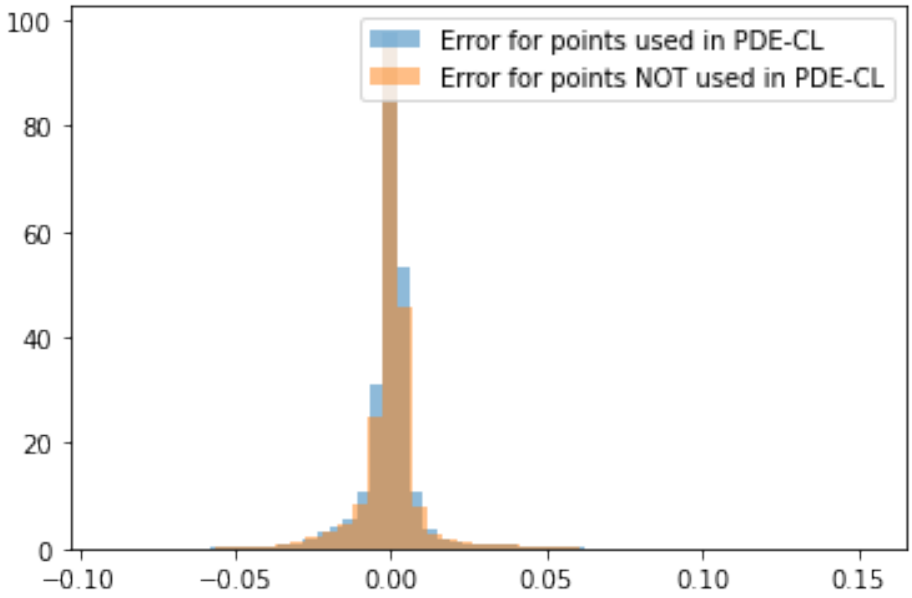
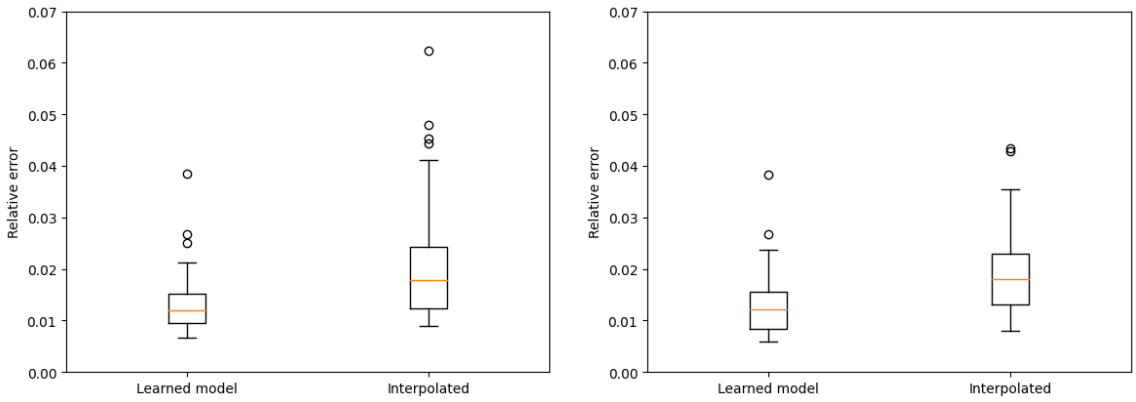


Figure C.4.1: **Histogram of errors: Error for points sampled by the PDE-CL, versus error for points not sampled.** We consider a trained model, and perform inference on a random PDE instance. In this plot, we consider the 1D convection setting. The histogram shows that the error for points used in the PDE-CL (1000 points) is about the same as error for points not used for the PDE-CL (9000 points). This demonstrates that we do not need to fit the PDE-CL on all points of the grid.

C.5 Ablation: Evaluating the quality of the learned basis functions

We implement an experiment to evaluate the quality (and the advantage) of our learned basis functions, compared to cubic interpolation. This experiment aims to understand whether our learned functions are useful outside of the points used for the constrained problem.

Problem setup. We start with a model trained on the 1D convection problem. The model was trained by sampling 750 points for fitting the PDE-CL, and 250 different points for the residual in the objective functions. The points were sampled from a 100x100 grid (10,000 points total). We sample 750 points from the grid, and solve the corresponding PDE-CL, which gives us a candidate PDE solution. For our baseline, We interpolate the solution we find on these 750 points to the 10,000 points using `scipy.optimize`'s cubic interpolation. We also interpolate to a 1000x1000 grid to see how our model's performance scales with resolution.



(a) 100x100 grid

(b) 1000x1000 grid

Figure C.5.1: **Quality of learned basis functions**. We compare the interpolation from the hard-constrained points against our model's learned prediction. Our learned basis functions have lower error, as compared to the baseline interpolation. The error gap increases with higher resolution on the grid of interest (i.e., a finer discretization). Our learned basis functions are 36% more accurate than the baseline interpolation for the 100x100 grid, and 37% more accurate than the baseline interpolation for the 1000x1000 grid.

Results. We compare the above results with the output of our trained hard-constrained model. Once the linear combination weights are fit (same as the baseline), we now use our learned basis functions to perform inference over all 10,000 (or 1 million) points. We plot our results over the test dataset in Figure C.5.1. The figure shows that using our model reduces the error, as compared to using a standard interpolation on the hard-constrained points.

C.6 Comparison to numerical solvers

We compare the complexity of our PDE-CL framework against numerical methods.

Problem setup. We define a $n_x \times n_t$ grid over a 1D domain with with n_x samples. We have a time horizon of $[0, T]$, with n_t samples. In this case, we assume that the PDE is linear. Let us suppose that we set the number of basis functions to N and the number of sampled points for solving the PDE-CL to $n = n_{interior} + n_{IC} + n_{BC}$. The PDE-CL will then solve a $(n + n_{IC} + n_{BC}) \times N$ linear system. Our method currently results in a dense linear system. Solving this linear system has complexity $O(\max(n_{interior} + n_{IC} + n_{BC}, N)^2 \times \min(n + n_{IC} + n_{BC}, N))$. This is added to a forward pass using the NN on the whole grid, once the optimal linear combination has been computed. Fortunately, this forward pass is embarrassingly parallel.

On the other hand, a finite difference method such as Crank-Nicolson or Lax-Wendroff requires solving a tri-diagonal system of size n_x at each t step. This yields an overall complexity of $O(n_x \times n_t)$. Comparing the complexity of both methods, the PDE-CL is asymptotically faster than numerical methods when,

$$\max(n, N)^2 \times \min(n, N) < n_x \times n_t. \tag{C.6}$$

In our current framework, inference is marginally slower or on par with numerical solvers. However, as we increase the resolution of our grid (finer discretization), our method compares favorably to the numerical solver—our computational cost increases more slowly than the numerical solver. Additionally, the operations in our PDE-CL are poorly optimized for current hardware, i.e., GPU utilization is low. Our method will greatly benefit from future improvements in hardware acceleration, which is still a nascent field in the context of linear solvers on GPUs.

Appendix D

Probabilistic forecasting with coherent aggregation

D.1 Details for each dataset

Tourism-Large. The Tourism-Large dataset [123] represents visits to Australia, at a monthly frequency, between January 1998 and December 2016. We use 2015 for validation, and 2016 for testing, and all previous years for training. The dataset contains 228 monthly observations. For each month, we have the number of visits to each of Australia's 78 regions, which are aggregated to the zone, state and national level, and for each of four purposes of travel. These two dimensions of aggregation total $N = 304$ leaf entities (a region-purpose pair), with a total of $M = 555$ aggregates in the hierarchy. We pre-process the data to include static categorical features representing the region, zone, state and purpose. We use a time-varying categorical feature representing the month of year (an integer between 0 and 11), and finally, the past observed values of the time-series.

Favorita. The Favorita dataset [137] contains grocery sales of the Ecuadorian Corporación Favorita in $N = 54$ stores. We perform geographical aggregation of the sales at the store, city, state and national levels, following [112]. This yields a total of $M = 94$ aggregates. Concerning features, we use past unit sales and number of transactions as historical data. We include several static categorical features provided by the dataset, such as item type categories, or precomputed store clusters. Finally, to capture seasonality, we use day of week and day of month categorical features.

Traffic. The Traffic dataset [127] contains aggregates of daily freeway occupancy rates for 200 sampled (out of 963) car lanes in the San Francisco Bay Area between January 2008 to March 2009. We follow the aggregation defined in Taieb and Koo [127]. We note that this scheme aggregates occupancy rates by adding them up. There are three aggregated levels: four groups of 50 car lanes, two groups of 100 car lanes, and an overall group of 200 lanes. Each group was chosen randomly in Taieb and Koo [127]; we keep the same

grouping. We follow previous experiments in the literature [127, 100, 112], and split the dataset into training, validation, and test dataset of size 120, 120 and 126. In Table 6.3, we report accuracy numbers for the last date of 126 dates only, following the experimentation setting in [100, 112].

Features

Tourism-Large. We describe the features in Table D.1.1.

Feature	Temporality	Kind
Observations	Past	Numerical
Month of year	Past/Future	Categorical
Region	Static	Categorical
Zone	Static	Categorical
State	Static	Categorical

Table D.1.1: Features used for the Tourism-Large dataset.

Favorita. We describe the features in Table D.1.2.

Feature	Temporality	Kind
Observations	Past	Numerical
Store-level total transactions	Past	Numerical
Day of month	Past/Future	Categorical
Day of week	Past/Future	Categorical
On-promotion indicator	Past/Future	Categorical
Family	Static	Categorical
Class	Static	Categorical
Type	Static	Categorical
Cluster	Static	Categorical
Store number	Static	Categorical
City	Static	Categorical
State	Static	Categorical

Table D.1.2: Features used for the Favorita dataset.

Traffic. We describe the features in Table D.1.3.

Feature	Temporality	Kind
Observations	Past	Numerical
Day of month	Past/Future	Categorical
Day of week	Past/Future	Categorical

Table D.1.3: Features used for the Traffic dataset.

Reported CRPS

For each dataset, CRPS is reported at different granularities, and then an overall CRPS is reported by taking a simple average across all levels. Below we discuss detailed definition of levels for each dataset, as shown in Table 6.3.

`Tourism-Large` Base time series in `Tourism-Large` represents number of visitors from 4 purposes and 76 regions, where regions can be aggregated up to zones, states and nation. After counting aggregates at different levels, there are 555 time series. Level 8 consists of average CRPS across all 4×76 purpose-region level predictions. Level 7 consists of average CRPS across 4×27 purpose-zone level predictions. Level 5 and 6, each consists of 4×7 purpose-state level predictions, and 4 predictions for all purposes at the national level, respectively. Similar to levels 5-8, CRPS for different geographical granularities are reported at level 1-4, but the predictions being evaluated are for number of visitors aggregated across purposes. For example, level 4 includes CRPS averaged across 76 regions, and level 1 is CRPS for national level prediction.

`Favorita` For 4K grocery items sold across 54 stores across 22 cities in 16 states in Ecuador, the average CRPS across 4036×54 item-store level predictions are reported at the "store" level. At the "city" level, average metric across 4036×22 item-city level predictions are reported. Similarly, we measure forecast accuracy at the "state" and "country" level.

`Traffic` We follow definition of hierarchies for `Traffic` data in [127]. 200 sampled car lanes are randomly aggregated to four quadrants, and further two halves, and lastly into one group as a whole. Level 4 in Table 6.3 includes average CRPS across 200 car lanes. Level 3 and 2 each consists of average CRPS across four quadrants and two halves. Level 1 consists of CRPS for the aggregated prediction for all car lanes.

D.2 Code Script for Sampling

We provide a snippet of code for sampling from our factor model in Figure D.2.1.

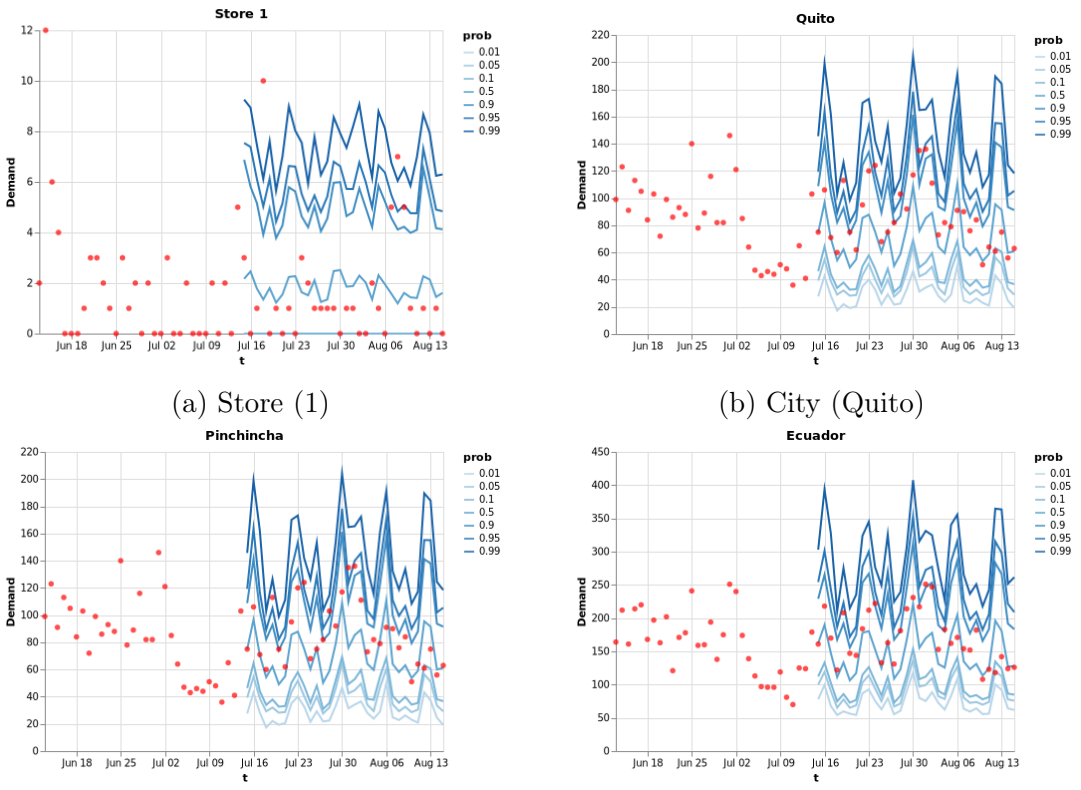
```
    def get_samples(factor_params, shares, stds, aggregation_mat,
                                    n_samples):
        concentration = factor_params[..., 0]
        rate = factor_params[..., 1]
        factor_samples = torch.distributions.Gamma(concentration, rate).
                                    rsample((n_samples,))
        samples = torch.distributions.LogNormal((shares * factor_samples).
                                    sum(-1), stds).rsample()
        aggregate = Aggregate(axis=2, ndim=6)
        return aggregate(samples, agg_mat)

```

Figure D.2.1: PyTorch function for sampling from our model, with a Gamma factor distribution and a Log-Normal base distribution. Note that the factor samples are shared across all base-level distributions. The samples are differentiable with regard to the function inputs. We can easily adapt this function to sample from other distributions. The parameters of the function are the outputs of a neural network.

D.3 Visualization of Predictions

In Figure D.3.1, we show our predictions for the `Favorita` dataset over a hierarchy containing Store 1, the city of Quito, the state of Pinchincha and the whole country of Ecuador.



(a) Store (1)

(b) City (Quito)

(c) State (Pinchincha)

(d) Country (Ecuador)

Figure D.3.1: Targets and predictions on the test set, for the hierarchy containing Store 1, for a given item, in the `Favorita` dataset. We visualize weekly forecast generated at the first forecast creation date in the test set. We show the forecasted quantiles at levels 0.01, 0.05, 0.1, 0.5, 0.9, 0.95 and 0.99 to demonstrate the spread of our forecasts, where the quantile forecasts are estimated empirically from 500 points from the factor model at each forecasted week. The model uses Gamma factors, and a Normal distribution clipped to be non-negative at the base-level. Clipping the Normal rather than truncating it allows to put point mass at zero, which is useful at the store level, as can be seen in Figure. D.3.1a: up to P10 quantile forecast is zero at the store level for this item for all evaluation weeks.