Title

Optimization Models and Methods for Large Scale and Complex Systems, with applications to distributed electricity resources integration

Author

Travacca, Bertrand

Publication Date

2021

Peer reviewed|Thesis/dissertation

Optimization Models and Methods for Large Scale and Complex Systems,
with applications to distributed electricity resources integration

by

Bertrand Travacca

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Scott Moura, Chair
Professor Kenichi Soga
Professor Laurent El Ghaoui

Summer 2021

Optimization Models and Methods for Large Scale and Complex Systems,
with applications to distributed electricity resources integration

Abstract

Optimization Models and Methods for Large Scale and Complex Systems,
with applications to distributed electricity resources integration

by

Bertrand Travacca

Doctor of Philosophy in Engineering - Civil and Environmental Engineering

University of California, Berkeley

Professor Scott Moura, Chair

This thesis focuses on models and methods for large scale and complex systems, with applications to distributed electricity resources. It is organized into three parts. In the first part, we show how to formulate the participation of distributed electricity resources – consisting of plug-in electric vehicles and residential photovoltaic panels – in the real-time and day-ahead electricity markets using convex optimization models. We show how these large scale problems can be efficiently solved by distributing them using a dual splitting approach. Using this structure, we then derive algorithms that allow to solve these problems efficiently and study their convergence properties. Finally, we illustrate our approach with multiple study cases on different markets. In the second part, we develop novel first-order heuristics methods, called Hopfield methods, based on Hopfield Neural Networks; in order to find candidate solutions to large-scale combinatorial optimization problems. We study the geometry and the convergence of these new methods and show how they connect with known convex optimization models and algorithms. We then illustrate how these methods perform on large nonlinear problems. In the last part, we present a new class of optimization models, implicit optimization, which includes deep learning, nonlinear control, and mixed-integer programming as special cases. Implicit optimization provides a unified perspective on these different fields, leading to new algorithms and surprising connections. We propose two tractable algorithms to solve such problems based on their implicit equation structure: implicit gradient descent and the Fenchel alternative direction method of multipliers. We illustrate our theory and methods with numerical experiments and dedicate a whole chapter on implicit deep learning architectures and methods.

I dedicate this thesis to the Oakland 'Castle in the Sky' community. They have been like family to me during these trying times.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to start by thanking my principal advisor: Professor Scott Moura, who has guided me throughout the entirety of my PhD. I would particularly like to thank him for the confidence he had in me from the start and the freedom he gave me to explore novel ideas during these five years of research. I am also grateful for the financial support I received from Total Energies. I want to thank the people working there who were actively involved in the successful completion of this project: Dr. Franck Ragot, Dr. Vincent Schachter, Dr. Carlos Carrejo, Dr. Brice Chung, Stanislas de Crevoisier, Dr. Gonzague Henri and Dr. Philippe Cordier.

The class in *Convex Optimization* thought by Professor Laurent El Ghaoui is the first one I took when I arrived at UC Berkeley in the fall of 2016. It is a team project with Sangjae Bae and Jiachung Wu from that class that ultimately led to the Chapter 1 (Part I) of this dissertation. The third chapter of Part I also started from a class project (*Energy Systems and Control* thought by Professor Scott Moura): among others, fellow students Greg Rybka and Kenji Shiraishi helped me understand the intricacies of the real time market in California.

In part II, The geometric interpretation idea for Hopfield methods comes from Professor Abhishek Hadler and Kenneth Paulo Caluya (from University of California Santa Cruz). For this part, I also received precious help from Mathilde Badoual for the coding implementation.

In part III, The Fenchel divergence and implicit deep learning ideas come from my co-advisor Professor Laurent El Ghaoui. In the last chapter on implicit deep learning, the simulations on MNIST and GTRSB were led by Fangda Gu. This chapter was co-written with Professor Laurent El Ghaoui, Armin Askari, Alicai Tsai and Fangda Gu. In the first chapter of part III, the application idea and simulation of the section titled *Solving overstay of plug-in electric vehicle charging with behavioral modeling* come from the work of Teng Zeng and Sangjae Bae.

Notation

Index notations

Given an integer $n \geq 1$ we denote by $[n]$ the set of integers between 1 and n (i.e. $1, 2, \cdots, n$). Given a vector $x \in \mathbb{R}^n$, we denote by x_i, $i \in [n]$, its components. Given $n_1 \geq n_2 \geq 1$, we denote the sub-vector of x with component indices between n_1 and n_2 as $x(n_1 : n_2)$.

Sequences

Given a sequence U, we denote the $k-$th element of the sequence with the exponent notation u^k. We use the following notation for the discrete derivative

$$\Delta u^k := u^{k+1} - u^k$$

Vectors

Given two vectors $x, y \in \mathbb{R}^n$, we denote the Euclidean scalar product of x and y by $x^\top y$ and we denote by $\|x\|$ the Euclidean norm of x. We will use the following notation for normalization, $n[x] := x/\|x\|$. We denote

$$\|x\|_p := \Big(\sum_{i=1}^{n} |x_i|^p \Big)^{\frac{1}{p}}$$

the p-norm, with $p \geq 1$. The maximum norm (limit case of the p-norm when $p \to \infty$) is given by

$$\|x\|_\infty := \max_{i \in [n]} |x_i|$$

We denote $\|x\|_S = \sqrt{x^\top S x}$ where S is positive semi-definite, the generalization of the euclidean norm to the geometry of S.

The notation $x \leq y$ refers to the element-wise inequality (i.e. $\forall i \in [n],\ x_i \leq y_i$). The elementwise product (also known as Hadamard product) between the two vectors is denoted by $x \odot y$. Similarly, we denote the componentwise division as $x \oslash y$. Given h, a real-valued function, $h(x)$ is a vector consisting of the map h applied component-wise to x. For instance,

if $x > 0$, $\log(x)$ is the logarithm applied to each component of x. We use the notation $x_+ = \max(0, x)$ for the ReLU function.

Matrices

We will denote by I the identity matrix, the size of which can be inferred in context. We denote $\mathbb{S}^n$ the set of symmetric matrices of size n and $\mathbb{S}^n_+$ the set of positive semidefinite matrices (i.e. $S \in \mathbb{S}^n$ such that all eigenvalues are positive). Alternatively, we will write $S \succeq 0$. We use the notation $S \succ 0$ to denote positive definite matrices (all eigenvalues strictly positive). For $S_1, S_2 \in \mathbb{S}^n$, we write $S_1 \succeq S_2$ to say that $S_1 - S_1 \in \mathbb{S}^n_+$. For $S \in \mathbb{S}^n_+$ we denote $S^{\frac{1}{2}}$ the matrix square root of S (i.e. the matrix M such that $M^2 = S$) .

For a matrix M, $|M|$ (resp. M_+) denotes the matrix with the absolute values (resp. positive part) of the entries of M.

$\|M\|$ refers to the operator norm defined by,

$$\|M\| = \max_{x \neq 0} \frac{\|Mx\|}{\|x\|}$$

We denote $\|M\|_\infty := \max_i \sum_j |M_{i,j}|$ the max-row-sum norm of M and similarly $\|M\|_1$ the max-column-sum norm of M. The Frobenius norm is defined by

$$\|M\|_F = \Big(\sum_{i,j} M^2_{i,j} \Big)^{\frac{1}{2}}$$

Any square, non-negative matrix M admits a real eigenvalue that is larger than the modulus of any other eigenvalue; this non-negative eigenvalue is the so-called *Perron-Frobenius eigenvalue*, and is denoted $\lambda_{pf}(M)$.
We will write the stacking of vector rows $v_1, \cdots, v_p \in \mathbb{R}^{1 \times n}$ as

$$V = [v_1; \cdots; v_p] \in \mathbb{R}^{p \times n}$$

Similarly, we will write the stacking of columns vectors $u_1, \cdots, u_n \in \mathbb{R}^p$ as

$$U = [u_1, \cdots, u_n] \in \mathbb{R}^{p \times n}$$

Maps and Convexity

Given a map f differentiable at a point x, we denote by $\nabla f(x)$ its Jacobian matrix (called gradient for real valued maps). The *convex conjugate* of f defined on a real topological space $\mathcal{X}$ is defined by

$$f^\star(v) := \sup_{x \in \mathcal{X}} (x^\top v - f(x)), \quad \forall x \in \mathcal{X}^\star$$

where $\mathcal{X}^\star$ is the dual space. Given a differentiable function f, we say that it is L-smooth iff

$$\|f(x) - f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathcal{X}$$

If f is twice differentiable, L-smoothness is equivalent to

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|, \quad \forall x, y \in \mathcal{X}$$

If f is convex and twice-differentiable, L-Smoothness is equivalent to

$$\nabla^2 f(x) \preceq LI, \quad \forall x \in \mathcal{X}$$

We refer the reader to [20] for a proof of equivalence between these statements. We say that f is l-strongly convex if it is convex and

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{l}{2}\|y - x\|^2, \quad \forall x, y \in \mathcal{X} \tag{1}$$

Alternatively, if f is twice differentiable, f is l-strongly convex iff

$$\nabla^2 f(x) \succeq lI \quad \forall x \in \mathcal{X}$$

We say that a function $f(x_1, x_2)$ is bilinear if given x_1, the map $f(x_1, \cdot)$ is linear and given x_2, the function $f(\cdot, x_2)$ is also linear. We say that a function $f(x_1, \cdots, x_m)$ is m-block multi-convex if given any $i \in [m]$, with all other variables fixed except x_1, the resulting function of x_i is convex.

Chapter 1

Introduction

This thesis focuses on optimization models and methods (algorithms) for large scale and (or) complex systems with applications to distributed electricity resources applications. We will start by placing these applications in energy systems in context as it is the source of motivation for the more theoretical research on heuristics for combinatorial optimization (Part II) and implicit optimization (Part III).

Arguably, the greatest challenge of this century is reducing the impact developed economies have on the climate of our planet. Green gas emissions can be attributed directly or indirectly to our energy use [74], and there are two complementary approaches to tackling this issue: (1) reducing energy use itself (2) and/or using *cleaner* energy – by cleaner we mean an energy usage that leads to less C02 tons equivalent emissions, currently evaluated with the life cycle analysis methodology [72].

Reducing energy use

Energy use reduction is often approached with technological innovation. For example the efficiency of cars running on gas has been improved during the last decades: with better engine combustion strategies, with a reduction of engine internal losses and rolling resistance (e.g. lighter vehicles) [79], or with the use of regenerative braking [24]. One limitation to technological innovation strategies is the so called *rebound effect*. In short, rebound effect occurs when the reduction of energy use is less than what is expected from a technological perspective. This phenomenon is often explained as follow: increased efficiency makes the use of a resource cheaper and therefore more affordable and more widespread [130]. Alternatively, rebound arises from a 'good conscience feeling': a resource might be used more because consumers believe that their behavior has a low polluting impact [151, 143]. For instance, an individual might use their electric car more than a car running on gas because they believe it has low impact. This is actually a bad example as research on mile traveled per electric vehicle suggest the opposite [28]. A better example being the gain in energy efficiency

of air travel during the last decades and its increasing impact on CO2 emissions [165, 48]. Some researchers argue that it is a rare phenomenon to see such new technologies inducing an immediate 'backfiring' rebound effect (i.e. rebound that results in even more energy consumption than previously observed) [2, 58]. Nevertheless, at a macro level –considering the long term implications and interactions with other technologies and impact on human behavior – backfiring is still an open question [130, 143].

These interrogations have lead to a different approach when it comes to reducing energy use: inducing a change in behavior to consume less energy. A recent example is the *flygskam* ('Flight Shame') movement that aims at discouraging flying to lower carbon emissions. To this day, this movement is mostly contained to Northern Europe [165, 56]. Such behavioral changes can also come from policies: for instance the UC Berkeley Institute of Transportation Studies has focused on understanding the choices being made when it comes to commuting, one of the aim being to find ways to favor public transportation or bike use [22, 65, 32]. Nevertheless, in the US, policies favoring directly less energy use are scarce, and when they exist they have an overall moderate impact [33]. As of today, the vehicles per miles traveled in the US has been increasing for over five decades [106]. The fact is that there is a strong linear correlation between GDP (i.e. wealth) and primary energy use [1, 70], and between primary energy use and the amount of goods and services being traded [166]. Given this fact, there has been limited interest in favoring policies (or funding research) aiming at reducing substantially energy use and consumption through systemic behavioral change in the US. This statement might seem contentious at first, we argue that it is a well known fact that in today's modern economic system, household consumption level (and therefore primary energy use), is one of the most important criterion used to assess the health of the economy. For instance, the fact that EV drivers drive half as less as other drivers seem to worry policymakers [28]. For these reasons, governments have favored a *decarbonized economy* strategy [97]: instead of using less energy, we use cleaner energy. For instance, one of the strategy of the US in that area is to electrify mobility [49] and increase the share of renewable energy production in the grid simultaneously [50]. Thereby achieving cleaner energy use. It is in that specific context that we should place our research in part I.

Cleaner energy use

When it comes to renewable energy integration, solar energy plays a central role, notably in California [29]. Contrary to non-renewable resources (at the exception of hydro and geothermal), it is not possible to control the power production capacity of renewable assets. Most of the time, it is what it is: we cannot control when the sun appears, when a cloud passes by or when the wind blows. The only controllability is curtailment, used at the expense of efficiency [164]. For instance curtailment is widely used in the wind energy industry: a notable example is China with its 50 TWh curtailment in that sector during the year 2016 [126] (for reference this figure matches the annual electricity consumption of Bangladesh and its 160 million inhabitants). Curtailment is also linked to the unpredictable nature of renewable electricity, another challenge for renewable integration in the grid. This problem

can be partly tackled with better prediction models and research on this topic has been blooming in the last few years [99, 107, 71]. An engineering solution to the controllability and predictability issues is energy storage. Research and development in energy storage consists in improving the technology itself or optimizing what can be done with the existing one. For instance, measuring more precisely the state of charge and state of health of a Li-Ion battery in order to inform better management strategies is still an active area of research [137, 111].

Distributed Energy Resources integration

One specificity of solar power is that it offers a convenient way of producing electricity at a local level (notably with rooftop solar [124]) and that it can go hand in hand with the electrification of mobility. Residential rooftop Photovoltaics (PV) associated with Plug-in Electric Vehicles (PEV), a form of electricity storage, charging is the local system we will focus on in the first chapters, and we will see how all these distributed systems can be coordinated to form a virtual power plant as in [120, 125, 132, 12]. Generally speaking, the goal of virtual power plants is to coordinate capacities of heterogeneous distributed energy resources (DERs) for the purposes of enhancing power generation. What do we mean by 'enhancing' in this context?

Many countries that have integrated a high level of solar power into their energy mix face the *duck curve* problem [81]: solar energy is produced during the day, if removed from the total demand (the result of this subtraction is often called *net energy* demand), the resulting power demand across the day displays a significant drop when the sun rises, and a significant increase when the sun sets. This phenomenon becomes more pronounced when more solar power is present. As a consequence, in order to follow this demand dynamic, more flexible generation is needed, which ultimately limits the integration of solar energy. Moreover, in order to follow these power ramps, costly (and polluting) gas power plants are often needed [155]. A solution to this issue is to flatten the duck curve with demand response: we can use the fleet of parked PEVs as a way to alter demand by deciding when to charge or not. Note that cars are parked approximately 95% of the time [15]. If we only consider all the Tesla cars sold in 2020, the total aggregated capacity of their batteries is approximately 3.5 GWh [150](for reference the maximum energy that can be produced in an hour by the Diablo Canyon nuclear power plant is 1GWh [43]). Of course, the 3.5 GWh we just mentioned is not an available energy resource per se: among others, PEVs, when parked are not always plugged to the grid and the average state of charge is often far from being zero (and vehicle discharge to grid is rare), nevertheless this rough figure gives an idea of the potential PEVs have when it comes to flattening the duck curve. Depending on the country or region, there are different ways for such demand response to be economically attractive. One such way is remuneration through electricity markets.

The electricity market purpose is the minimization of the cost for supplying a given amount of energy during a given time period. The way markets operate is country and region specific. Nevertheless, the Day Ahead Market (DAM) exists and operates in a very similar ways in many countries across Europe and in the United States: one day before

operation, bids (quantity and price) for electricity supply and demand for each hour of the next day are made. The market operator then proceeds by clearing the bids that minimize the cost of operation for the whole system (sometimes considering congestion and losses in the grid). The DAM cleared prices reflect the duck curve indirectly: most of renewable energy production have zero marginal cost, which tends to mechanically lower the prices during the day, while ramping up or down electricity production results in higher price because only a portion of generators are able to provide quick variations of power (e.g. gas power plants). Moreover prices are high because these flexible generators remunerate themselves only during these events. Therefore as an objective for operating Distributed Energy Resources (DERs), composed of PEVs and distributed PV generation, we will consider participation in the DAM, achieving two purposes that go hand in hand: facilitating the integration of renewable energy by flattening the duck curve and making the system more efficient by reducing the cost of operation.

We mentioned earlier that one of the issues with renewable energy integration is unpredictability. More generally speaking, there are many uncertainties at play in the electricity system [174]: demand is unpredictable, generators can experience failures, and so does the grid. Therefore, there are other mechanisms in place to align supply and demand during the lapse of an hour: ancillary services and real time markets (RTM) among others. Participation of DERs in these markets can therefore also alleviate the unpredictable nature of renewable energy. Most research on distributed energy resource participation in RTMs do not take into account its operation specificities. In the last chapter of Part I, our main contribution is to show how we can formulate the coordination of DERs as a convex optimization problem, considering when information is available and when a decision can be made.

Distributed optimization

As a start, in the first chapter of part I, we show how we can formulate the participation of DERs (PEVs and rooftop PV) in the DAM as a convex optimization problem and present how the computational burden can be distributed to local agents, while limiting the exchange of private information to a central entity. Distributed convex methods have been used for coordinating DERs previously in [36, 105, 94, 94], our main contributions in that application setting are as follow:

- When bidding on electricity markets we do not consider that we know the cleared prices beforehand. Instead we build a prediction and a statistical model on the error. More precisely, we estimate a covariance matrix that we use to take into account the risk of our prediction using a Markowitz portfolio approach [26].
- We consider uncertainty on the aggregate state of DERs within our perimeters (such as the average state of charge and mobility use).
- We provide a detailed convergence analysis for our distributed algorithms and the specific structure of the problem: the uncertainty consideration on mobility leads to a

sub-linear convergence proof. We actually show that this analysis is more pertinent than research displaying linear convergence when the problem scales up.

- We point out the inherent limitations of some distributed methods when the number of agents increases and propose the use of distributed algorithms using a combination of stochastic methods and Nesterov acceleration techniques [114]. We also provide a convergence rate analysis for these new algorithms.

In this firs part, as it is done in most research on PEV charge scheduling, we model the charging decision as a continuous variable: at a given point in time we assume that we can charge at a power level that can be continuously chosen between 0 and 10kW (for example). In practice most PEV chargers only allow for either charging – possibly at different power levels – or not charging [133]. This consideration leads to mixed integer or combinatorial optimization problems (NP-hard and non-convex [96]). In the particular case of distributed optimization, where agents solve their local problems, this might not be an issue if the local problems are not high dimensional. For one PEV participating in the DAM per prosumer as in Chapter 2, the decision variable would be in $\{0,1\}^{24}$ (for each hour of the day decide whether to charge or not). A problem of this size can be solved locally using combinatorial optimization solvers such as CPLEX by IBM [23] in less than a second. Nevertheless, for combinatorial problems in $\{0,1\}^n$ with $n > 50 - 100$ such solvers can become too slow. This threshold is already reached if we consider two PEVs simultaneously. Therefore, we see that even if the problem can be distributed, solving local problems is a bottleneck. Particularly given the fact that the local problems need to be solved many times in order to reach a consensus. We remark that most commercial combinatorial problem solvers try to find the optimum, but for many problems that arise in energy scheduling, sub-optimality is not an issue. For that reason, we decide to explore and revisit Hopfield Neural Networks, a first order method heuristic for combinatorial problems introduced in 1985 by Hopfield & Tank.

Combinatorial Optimization and Hopfield Neural Networks

Hopfield neural network consists in the time evolution of a state and a hidden state, in which the hidden state dynamic is linear with respect to the state and the state itself obtained by applying a non-linear activation map to the hidden state [76, 77]. Even though HNNs original purpose was not to solve combinatorial problems (it served as a content addressable memory [76]), in [78], the authors notice that its dynamic could be used to build an algorithm for the traveling salesman problem. It was shown during the following years to be a good heuristic for many combinatorial problems. Seeing the potential of HNNs a a tool to solve problems in energy systems (among others), we decide to revisit and extend them using a modern approach. Our contributions are as follow,

- we extend HNNs to non quadratic objective and propose a hidden state evolution that is nonlinear with respect to the state.

- we show that HNNs can be interpreted as a gradient descent on a non-euclidean geometry, called mirror descent and provide a geometric interpretation for HNNs as a heuristic to solve combinatorial problems.
- we extend HNNs to more general descent methods that can improve the quality of the heuristic and we propose an algorithm for finding a descent direction at each iteration similar to that of the Frank-Wolfe algorithm.
- We advance the convergence analysis for HNNs by proving not only numerical convergence of the objective across iterations but also a convergence rate of $O(1/k)$ similar to that of convex methods.
- We propose a dual method approach allowing the consideration of linear constraints. Hence, the final method we build is a heuristic for solving nonlinear combinatorial problems under linear constraints.

We implement our extended Hopfield methods on random combinatorial and nonlinear problems. We also lead a study case for optimal economic load dispatch where generators can be turned on or off (with an associated cost to such actions). We show through these experiments, comparing our solutions to exact solvers or convex relaxation heuristics, that the Hopfield method offers good solutions in many instances. We believe that there is a potential for the use of such methods across a variety of problems that arise in energy systems where some of the optimization variables are discrete and when the problem is too large to use exact solvers.

To this day a stochastic version of Hopfield methods, called Boltzmann machine is still widely use as an element of deep learning architectures for unsupervised learning [62] where it used to evaluate probability distributions on datasets. As we mentioned earlier, one of our contribution was to show that HNNs are a special case of mirror descent, mirror descent can be seen as a proximal algorithm using a Bregman divergence (instead of an L2 regularization).

What about other forms of divergence? It is that question that led to the idea of implicit optimization in part III. In this last part, we introduce the Fenchel divergence, a mapping that is able to represents whether a constraint of the form $x = \phi(y)$ is satisfied. Interestingly, this type of constraint is present by definition in Hopfield methods. The Fenchel divergence, via the Fenchel conjugate is also directly related to duality theory, and therefore offers connections to distributed optimization.

Implicit optimization

Implicit optimization offers even more surprising connections regarding our research in distributed optimization, HNNs and combinatorial optimization. Implicit optimization corresponds to optimization programs with constraints that are implicit, meaning that part of the optimization variable is defined implicitly via an equation of the type $x = \phi(x, u)$ with ϕ a nonlinear map we call *implicit map*. We consider two possible conditions for this map: a

contraction condition or what we call the Fenchel condition (in reference to the Fenchel-Young inequality [26]). This type of optimization modeling being new, our first contributions are:

- showing, via the universal approximation theorem [30, 40], the generality of the Fenchel condition, mainly the fact that any nonlinear map ϕ approximately satisfies it.
- we show that implicit models can be used for a multitude of applications and encompasses a multitude of optimization problems across different fields. Notably, we show that we can use it: (1) to represent combinatorial optimization problems, (2) to solve nonlinear control problems and (3) learn with neural networks.
- we show how to compute the gradient using the implicit function theorem and propose a stochastic approach.

This new modeling approach also leads to novel methods, and therefore new ways of approaching combinatorial optimization, nonlinear control, learning with neural networks. We explored and developed two sets of methods for implicit optimization:

- in some cases, gradient descent can be run in a tractable manner. We show that this is the case under the contraction condition via the use of the implicit function theorem.
- a novel method called Fenchel Alternative Direction Method of Multipliers (Fenchel ADMM). Which is an algorithm that uses duality and an augmented Lagrangian with Fenchel divergence penalization (in lieu of the classic L2 penalization). We show that each step of the algorithm consists of convex optimization problems.

We remark that dual methods are at the core of distributed methods, this method can be used to coordinate DERs with nonlinear constraints tying them implicitly. After illustrating our methods, (1) by solving random quadratic combinatorial optimization program, (2) with a study case on PEV charging parking management with behavioral modeling; we devote a whole chapter on implicit deep learning. We show that implicit deep learning, which can be seen as an infinite sequence of feedforward networks with the same weight at each level, contains most known deep learning structures. Our contributions on that subject are as follow:

- We establish rigorous and numerically tractable sufficient conditions for the implicit deep learning model to be well-posed and show how such models can be composed to form even more intricate and rich models
- we show how to model a variety of network architectures (fully connected, convolutional networks, residual and recurrent networks) as implicit models

Finally, we illustrate our new models and methods on (1) synthetic data as well as real datasets such as (2) MNIST and the (3) German traffic sign recognition dataset.

All in all we believe that with accelerated distributed methods, Hopfield methods and implicit optimization; we have advanced modeling capabilities and methods for energy systems applications. With accelerated distributed methods we show how we can coordinate a large fleet of DERs to facilitate renewable electricity integration, with Hopfield methods we have a heuristic for large combinatorial problems that can arise when modeling systems such as interconnected PEVs. With implicit optimization we propose new methods to tackle non-linearity in the constraints (e.g. that can arise when considering a behavioral model or a nonlinear model for a battery) and also solve combinatorial problems. We branch out with implicit deep learning, a new class of deep learning, that is not directly connected to energy applications, but that could be extended, for instance, to form a new approach to reinforcement learning (a technique widely used today in energy systems [93, 117, 86]) or simply as a tool for learning from energy datasets, may it be forecasting weather conditions [161] or predicting electricity prices.

Part I

Chapter 2

Distributed Convex Optimization Methods for DER Participation in Electricity Markets

In this chapter we present an optimal Day-Ahead Electricity Market (DAM) and bidding strategy for an aggregator leveraging a pool of residential prosumers: residential customers with local photovoltaic (PV) production and plug-in electric vehicle (PEV) charging flexibility. The goal of the aggregator is to optimally manage its pool of flexible resources, in order to minimize its cost given these two markets.

The aggregator's point-of-view differs from the social planner angle that is often taken in the existing literature: mainly the aggregator is considered to be a private entity (e.g. an electricity retailer). We propose a novel approach to tackling this optimization problem by including risk management in the objective function and chance constraints on the aggregated PEV mobility constraints. In a first step, we model local system constraints and define a stochastic optimization scheme that exploits the problem structure to distribute the objective among prosumers via dual-splitting. Dual splitting is achieved with two consensus variables: shadow prices for energy and for PEV charging. In a second step, we propose a projected gradient ascent algorithm to solve the dual problem and prove a rate of convergence.

In the first part of the chapter, we will focus largely on the day-ahead market (DAM) to illustrate our methods. In Chapter 4 we will show how to model and incorporate a real time market (RTM) strategy using Model Predictive Control (MPC).

2.1 Introduction

Context and Motivation

As a global leader in climate change policy, California has one of the highest renewable energy target in the world; 50% of electricity supply will be from renewable electricity sources (RES)

by 2030. [1] Most of the RES that California uses are variable and intermittent such as wind and solar power. The need to ramp up or ramp down controllable, often non-renewable, generation sources due to the variability of the RES is a current and increasing challenge for power system operators and policy makers.

Historically, supply-side resources have followed the load. Load, in general, has been considered to be inelastic, as the prices that are observed by retail customers do not correspond to wholesale electricity prices. End-users are most of the time subject to flat rates. With the increase in variable supply resources there is increasing interest in the use of demand-side resources to participate in the wholesale markets as a mechanism to resolve some of the supply-demand imbalances that could otherwise occur.

In this context, both demand response and energy storage are considered possible mitigating measures and technologies that can provide flexibility to the grid. In many electricity markets (e.g. Germany, Italy, California) peak demand occurs after sunset, when solar power is no longer available, this phenomenon is commonly referred to as the *duck curve* [42] . Peak power demand creates a need for more flexible power supply [80] which could be leveraged on the residential demand-side as suggested in [36, 4, 169].

We take the example of the California Independent System Operator (CAISO) that operates three distinct markets, a day-ahead market, a real-time market, and ancillary services (such as congestion revenue rights and convergence bidding).

In this chapter, residential end-users with controllable plug-in electric vehicle (PEVs) chargers and photovoltaic (PV) systems are referred to as prosumers. An aggregator is a company pooling prosumers to bring them to the Day-Ahead energy Market for electricity (DAM). The DAM takes place one day before the operating day, and consists in various market entities bidding prices and quantities for each hour of the next day. The objective of the aggregator is to maximize the total electricity profits it delivers in the markets. It is important to highlight the fact that a single prosumer cannot participate in electricity markets because it does not fulfill the power threshold requirements of the market regulator. It is true that Time-of-Use tariffs (TOU) and Real Time Prices (RTP) aim to bring the market to the prosumer level. Nevertheless, there might arguably be at this level a low acceptance of RTP [7]. Moreover TOU rates are not able to capture entirely the state of the electricity system and can even create higher peak loads as [131] suggests.

Two main challenges for managing a large population of flexible resources are (i) uncertainty and (ii) computational scalability. (i) First, we must ensure consumer comfort (e.g. mobility) in the face of uncertain electricity consumption and limit financial risk in the face of uncertain DAM prices. (ii) Second, we require computationally scalable scheduling algorithms that guarantee delivered power from the aggregator to the power system operator.

[1] In early 2017, California Senate leader Kevin de León put forth a bill that would mandate the State to use 100% renewable power by 2045.

Literature Review

A growing body of literature addresses optimal charging of PEV populations and residential demand response [36, 105, 94, 94]. This research can be classified as having centralized or distributed protocols. Centralized algorithms [37, 144, 95], use a central infrastructure to communicate with each agent, gather their information, and compute the optimal aggregated load profile. The challenges for centralized methods are scalability with respect to communication and computation, as well as privacy issues. In distributed or decentralized optimization algorithms [94, 129], each local agent solves its own problem and communicates information to its neighbors or the aggregator.

Market bidding strategies and market uncertainty for aggregated PEVs have been studied in [156, 157, 21]. The aforementioned methods could successfully address uncertainty in aggregated load scheduling, but do not provide a rigorous convergence analysis (except [94]). In particular, finding the necessary number of iterations to reach a specific precision is crucial if we seek to assess implementation burdens for the aggregator.

In this chapter, we construct a tailored optimization method for scheduling uncertain electricity resources in the uncertain DAM using aggregated resources. Leveraging the particular structure of the problem, we derive a distributed dual-optimization scheme. We then perform a convergence analysis to yield an explicit upper-bound on the rate of convergence, for the projected gradient ascent algorithm. Finally, a case study is implemented to illustrate the performance of our algorithm. Ultimately, the contributions of this chapter to ensemble DER control are twofold:

1. Formulation of a convex optimization scheduling problem and distributed algorithm that accounts for uncertain DAM and RTM prices and PEV availability.

2. Proof of an upper-bound for convergence

Chapter Structure

The report is structured as follows:

- In Section 2.2, we formulate the DER model with local constraints on power, energy, and availability. Next, we define an optimization model to address risk management for DAM bidding, in the face of uncertain DAM prices and uncertain PEV mobility aggregation.

- In Section 2.3, we show how to exploit the problem structure to enable distributed computations.

- In Section 2.4, we propose a distributed gradient ascent method to solve the problem and derive an explicit bound for convergence.

- In Section 2.5, we illustrate our model for price prediction and corresponding risk (covariance matrix). We then show how our algorithm performs with a case study of 100 prosumers.

Notation and Nomenclature

The following notation is specific to this chapter. Uppercase letters refer to variables with units of power (kW) while lower case letters refer to variables with units of energy (kWh). Symbol $x(t)$ refers to the value taken by variable x at time t. In the absence of the exponent we will consider the variable x as a vector with $x(t)$ as its components. The index x_i refers to a local prosumer variable $i \in [N]$ and $\mathbf{sum}(x) = \sum_{i=1}^{N} x_i$ to the sum of these local variables. Symbol $\hat{X}$ refers to the average or estimate of a random or unknown variable X. Finally, $\overline{x}$ (respectively $\underline{x}$) refers to an upper (lower) bound of the variable x. In the following table we gather the nomenclature that will be used in this chapter. These notations will be progressively introduced, but the reader can come back to this table at any point for a quick look up.

2.2 Local System Model and Problem Formulation for Day Ahead

We start by building the model used in the DAM, we will come back to this local prosumer model once we tackle the RTM in chapter 4. We will consider that each prosumer has a PEV, a PV installation and is connected to the grid.

Local Model for the Prosumer

Let us consider a given prosumer $i \in [N]$, with N the number of prosumers.

Local Power Balance

At any point in time the local electricity demand, composed of the residential load L_i and the charging rate of the PEV (EV_i), has to be met with supply either with the local PV installation (S_i) or power from the grid (G_i). We have

$$L_i + EV_i \leq S_i + G_i \tag{2.1}$$

Where the inequality here should be interpreted elementwise. In other words, this inequality must hold at any point in time. Note that the power balance has been relaxed from an equality constraint to an inequality constraint representing the fact that PV production can be curtailed.

Table 2.1: Nomenclature for Part I

N	Number of prosumers
Δt	Time-step for the Day Ahead Market: 1 hour
Δt_{RT}	Time-step for Real Time Market: 0.25 hours (15 min)
T	Time-Horizon (hours)
T_H	Model Predictive Control Time-Horizon for RTM (hours)
ρ or ρ_{DA} and ρ_{RT}	respectively DA and RT risk aversion parameters
δ	L2 regularization parameter
	All of the following variables are $\in \mathbb{R}^T$
p or p_{DA}	Day-Ahead Market price
L_i	Uncontrollable residential load of prosumer i
S_i	Solar PV production of prosumer i
EV_i	Day Ahead Charging rate of PEV i
ev_i	State of Energy of PEV i
G_i	Day Ahead Power imported from the grid for prosumer i
	All of the following variables are $\in \mathbb{R}^{4T}$
p_{RT}	Real-Time Market price
ΔEV_i	Real Time Charging rate of PEV i deviation from day ahead schedule
ΔG_i	Real Time Power imported from the grid for prosumer i deviation from DA
C	covariance matrix for DAM price prediction error, $\in \mathbb{R}^{T\times T}$
C_{RT}	covariance matrix for RTM price prediction error, $\in \mathbb{R}^{4T_H\times 4T_H}$

Local Grid Constraints

At any given node in the distribution network, there is a limit on power import or export (2.2). Typically, for residential customers, the power import cap from the grid is $\overline{G}_i \simeq 10\,kW$

$$\underline{G}_i \leq G_i \leq \overline{G}_i \tag{2.2}$$

Note that our model makes it possible to export power to the grid (i.e $\underline{G}_i < 0$).

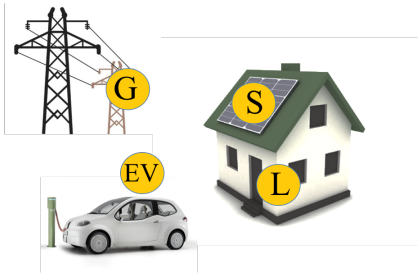

Figure 2.1: Prosumer system representation

Local PEV Constraints

We use a simple model of charge for the PEV: we consider perfect round-trip efficiency for the battery. Equation (2.3) governs the PEV battery state of energy (SOE) dynamics. The difference with the model used in [94] is the fact that the PEVs are not in a closed system. When the i-th PEV leaves its location, it is no longer in the aggregator's perimeter. For instance, a PEV can leave with a half state of charge and come back fully charged (because it has charged elsewhere). Therefore, EV_i represents the on-site charge only, while $EV_{D,i}$ is a value that represents the observed and uncontrolled charge or discharge of the PEV while off-site. We denote ev_i the state of charge of a PEV for prosumer i, we have the following dynamic for the charge of the battery

$$ev_i(t) = ev_i(t-1) + EV_i(t)\Delta t - EV_{D,i}(t)\Delta t \tag{2.3}$$

We consider that the state of charge at each given point in time should be between a minimum and maximum state of charge (denoted $\underline{ev}_i(t)$ and $\overline{ev}_i(t)$ respectively), we rewrite (2.3) as an on-site cumulative energy consumption constraint, which is expressed as:

$$\underline{ev}_i := \underline{ev}_i + AEV_{D,i} \leq AEV_i \leq \overline{ev}_i := \overline{ev}_i + AEV_{D,i} \tag{2.4}$$

Where we incorporate in the bounds of the state of charge the uncontrolled charge and discharge of the PEV while keeping the same notation. In the previous equation, A is the discrete integration matrix:

$$A := \Delta t \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{T \times T}$$

Finally, the PEVs charging power constraint is given by:

$$\underline{EV}_i \leq EV_i \leq \overline{EV}_i \tag{2.5}$$

Note that when the PEV is un-plugged at a given time t, we require $\underline{EV}_i(t) = \overline{EV}_i(t) = 0$ because the battery is not physically present. For the purpose of conciseness, the convex feasibility set generated by the local constraints (2.1), (2.2), (2.4) and (2.5) is hereby referred to as $\mathfrak{L}_i$.

Optimization problem formulation

Aggregator's objective

We model the DAM price p as a random variable in $\mathbb{R}^T$ that follows a multivariate Gaussian distribution

$$p \sim \mathcal{N}(\hat{p}, C), \text{ with } C \succ 0$$

The total power bought or sold by the aggregator in the DAM is the vector $\mathbf{sum}(G) = \sum_{i=1}^{N} G_i$. The choice of the quantity imported from (or exported) the grid $\mathbf{sum}(G)(t)$ for $t \in [T]$ can be considered as a portfolio problem where the assets are the DERs, the returns are the DAM prices and the budget constraint is the linear flexibility constraints $\mathfrak{L}_i$ described previously. The optimization of $\mathbf{sum}(G)$ involves a trade-off between the expected DAM cost $\hat{p}^\top \mathbf{sum}(G)$ and its variance $\mathbf{sum}(G)^\top C \mathbf{sum}(G)$.

We define the optimization objective f for the aggregator to be given by (2.6), With ρ representing the risk aversion propensity of the aggregator and δ a regularization parameter. We aim at producing a *Pareto optimal* choice that balances the expected cost/benefits and the risk that comes from making the DAM price prediction $\hat{p}$

$$f := \hat{p}^\top \mathbf{sum}(G) + \frac{\rho}{2} \mathbf{sum}(G)^\top C \mathbf{sum}(G) + \mathcal{R}(EV, G) \tag{2.6}$$

with the regularization

$$\mathcal{R}(EV, G) := \sum_{i=1}^{N} \mathcal{R}_i(EV_i, G_i) := \frac{\delta}{2} \sum_{i=1}^{N} \|EV_i\|^2 + \|G_i\|^2 \tag{2.7}$$

This regularization term penalizes PEV battery degradation as described in [94], [112]. Nevertheless, battery degradation can also depend on other factors than charging power magnitude. The regularization in $\mathbf{sum}(G)$ penalizes the magnitude of the G_i's and can be interpreted as a cost linked to local stability of the distribution grid.

Aggregated PEV mobility uncertainty

Predicting PEV mobility at the local level is not tractable, as the local PEV use is highly unpredictable and it may require access to sensitive private information. This is not the case at the aggregated level, where local data is not needed and behaviors are *smoothed* via the law of large numbers. Therefore we make the choice of modeling PEV mobility and its uncertainties at the aggregate level via chance constraints – explained next.

We hypothesize that the aggregated energy and power bounds, that we denote:

$$\underline{\mathbf{sum}(ev)}, \overline{\mathbf{sum}(ev)}, \underline{\mathbf{sum}(EV)}, \overline{\mathbf{sum}(EV)}$$

follow multivariate normal distributions with diagonal covariance matrices. Remark that for N large, the central limit theorem supports part of this hypothesis (nevertheless there is no guarantee that the covariance matrix is diagonal). We require these bounds to be respected for each hour with a probability of at least η. Let us denote $\Phi(\cdot)$ the cumulative distribution function for a zero-mean, unit variance Gaussian random variable. Consider the first lower bound. We can write

$$\underline{\mathbf{sum}(ev)}(t) \sim \mu(t) + \sigma(t)\mathcal{N}(0,1) \tag{2.8}$$

where $\mu(t)$ and $\sigma(t)$ are the mean and standard deviation. Instead of requiring this lower bound to be satisfied for all realizations, we require that the lower bound is satisfied with probability η,

$$\mathbb{P}\Big(\underline{\mathbf{sum}(ev)}(t) \leq (A\mathbf{sum}(EV))(t)\Big) \geq \eta \tag{2.9}$$

which is equivalent to

$$\big(A\mathbf{sum}(EV)\big)(t) \geq \mu(t) + \sigma(t)\Phi^{-1}(\eta) := \underline{\mathbf{sum}(ev)}_{\eta}(t) \tag{2.10}$$

The same procedure is applied to the other aggregated random lower and upper bounds. The use of chance constraints allows to be more *robust* to possible prediction error on mobility when optimizing for G. In this section, we presented some linear equality constraints to be considered for robustness purposes. These type of constraints could also be considered for the grid stability. For instance consider a neighborhood with 100 prosumers connected to a single substation with a $500kW$ power limit: in that scenario all the prosumers cannot reach their $\overline{G}_i = 8kW$ power cap at the same time. To guarantee the stability of the network, we could consider the constraints,

$$G_i \leq 8 \text{ kW } \forall i \in [N] \text{ and } \mathbf{sum}(G) \leq 500\text{kW}$$

Which is similar in structure to the robust constraints we are considering.

Optimization Problem Formulation

We are now positioned to formulate a *robust* optimization problem for the aggregator

$$\begin{aligned}\min_{EV_i, G_i, \mathbf{sum}(EV), \mathbf{sum}(G)} \quad & p^\top \mathbf{sum}(G) + \frac{\rho}{2}\mathbf{sum}(G)^\top C \mathbf{sum}(G) + \mathcal{R}(EV, G) \\ \text{s.to:} \quad & \text{Local constraints} : \forall i \;\; \mathfrak{L}_i \\ & \text{Aggregated variable constraints} : \\ & \mathbf{sum}(EV) = \sum_{i=1}^{N} EV_i, \quad \mathbf{sum}(G) = \sum_{i=1}^{N} G_i \\ & \textit{Aggregated EV (robust) constraints} : \\ & \underline{\mathbf{sum}(ev)}_\eta \leq A\mathbf{sum}(EV) \leq \overline{\mathbf{sum}(ev)}_\eta \qquad (2.11) \\ & \underline{\mathbf{sum}(EV)}_\eta \leq \mathbf{sum}(EV) \leq \overline{\mathbf{sum}(EV)}_\eta \qquad (2.12)\end{aligned}$$

We denote $f^\star$ the optimal cost for (2.12). This optimization problem couples the local charging and grid import/export (EV_i, G_i) optimization variables in the objective via the variance term $\mathbf{sum}(G)^\top C \mathbf{sum}(G)$. Additionally, coupling terms appear with the aggregated robust PEV mobility constraints. Without these coupling terms, the optimization problem is sum-separable (meaning that each local prosumer can solve its own optimization problem). That said, the problem contains independent local constraints $\mathfrak{L}_i$ as well as a sum-separable regularization and objective term

$$\hat{p}^\top \mathbf{sum}(G) = \sum_{i=1}^{N} \hat{p}^\top G_i$$

This structure can be exploited to distribute the objective – discussed next.

2.3 Distributed Optimization Scheme

Dual splitting can be used to exploit the problem structure the aggregator faces. We show in the following theorem that (2.12) is equivalent to a dual problem

Theorem 2.3.1. *Solving (2.12) is equivalent to solving:*

$$\max_{\substack{\mu, \nu \\ \textit{s.to: } \mu \geq 0}} \left(-\frac{1}{2\rho}\nu^\top C^{-1}\nu + c^\top \mu + \sum_{i=1}^{N} \min_{\substack{EV_i, G_i \\ \textit{s.to: } \mathfrak{L}_i}} \left(G_i^\top (p - \nu) + EV_i^\top B\mu + \mathcal{R}_i(EV_i, G_i) \right) \right)$$

With $B \in \mathbb{R}^{T \times 4T}$, and $c \in \mathbb{R}^{4T}$ given parameters (cf. Proof).

Proof. This proof uses duality theory: namely, the aggregated decision variables and chance constraints from (2.12) are used to form the dual problem and the corresponding Lagrangian $\mathcal{L}$ given by (2.13). Slater condition holds, henceforth the min and max symbols can be interchanged without generating a dual gap.

$$\begin{aligned}\mathcal{L} = &\sum_{i=1}^{N} \left(p^\top G_i + \mathcal{R}_i(EV_i, G_i)\right) + \frac{\rho}{2}\mathbf{sum}(G)^\top C\mathbf{sum}(G) \\ &+ \mu_1^\top(\underline{\mathbf{sum}(ev)}_\eta - A\mathbf{sum}(EV)) + \mu_2^\top(A\mathbf{sum}(EV) - \overline{\mathbf{sum}(ev)}_\eta) \\ &+ \mu_3^\top(\underline{\mathbf{sum}(EV)}_\eta - \mathbf{sum}(EV)) + \mu_4^\top(\mathbf{sum}(EV) - \overline{\mathbf{sum}(EV)}_\eta) \\ &+ \nu_1^\top(\mathbf{sum}(EV) - \sum_{i=1}^{N} EV_i) + \nu_2^\top(\mathbf{sum}(G) - \sum_{i=1}^{N} G_i) \end{aligned} \tag{2.13}$$

where the μ's≥ 0 and ν's are the dual variables associated respectively with the inequality and equality constraints. Using the fact that

$$\min_{\mathbf{sum}(G)} \frac{\rho}{2}\mathbf{sum}(G)^\top C\mathbf{sum}(G) + \nu_2^\top \mathbf{sum}(G) = -\frac{1}{2\rho}\nu_2^\top C^{-1}\nu_2$$

And the following equality constraint on the dual variables that ensures the problem is bounded

$$A^\top(\mu_2 - \mu_1) + \mu_4 - \mu_3 + \nu_1 = 0$$

Leads a formulation for the dual function

$$g(\mu, \nu) = -\frac{1}{2\rho}\nu^\top C^{-1}\nu + c^\top\mu + \sum_{i=1}^{N} \min_{EV_i, G_i,\ \mathbf{s.to.}\ \mathfrak{L}_i} G_i^\top(p - \nu) + EV_i^\top B\mu + \mathcal{R}(EV_i, G_i)$$

with $\mu = [\mu_1;\ \mu_2;\ \mu_3;\ \mu_4]$, $\nu = \nu_2$, $c = [\underline{\mathbf{sum}(ev)}_\eta;\ -\overline{\mathbf{sum}(ev)}_\eta;\ \underline{\mathbf{sum}(EV)}_\eta;\ -\overline{\mathbf{sum}(EV)}_\eta]$ and

$B = \begin{bmatrix} -A^\top & A^\top & -I & I \end{bmatrix}$

Which concludes the proof. □

Remark 2.3.1. *Dual variables are often interpreted as* shadow prices. *A shadow price corresponds to a monetary value assigned for a constraint to be enforced. Here the term $-\nu^\star$ can be interpreted as a shadow price for power export/import and $B\mu^\star$ as a shadow price for PEV charging.*

Theorem 2.3.1 shows that only the dual variables $(\mu^\star, \nu^\star)$ are needed for the prosumers to reach a consensus equivalent to solving the primal problem (2.12). With this distributed scheme the aggregator can partake the primal objective and the computational burden between the prosumers (each prosumer solves its own linear constrained convex quadratic program). In the next section, we discuss an algorithm to find the optimal dual variables $(\mu^\star, \nu^\star)$ and converge to a consensus among prosumers.

2.4 Distributed Ascent Method

In this section, a projected gradient ascent algorithm with constant step-size is outlined to solve the Lagrange dual problem in Theorem 2.3.1. The algorithm's rate of convergence and its so-called *oracle complexity* $K(\varepsilon)$ are derived. The oracle complexity guarantees that for a number of iterations $k > K(\varepsilon)$:

$$|f^\star - g(\mu^k, \nu^k)| < \varepsilon \tag{2.14}$$

$\varepsilon > 0$ is the desired solution precision, $\mu^\star$. In practice we do not have access to the value $f^\star$ in advance. Therefore the criterion (2.14) is not practical – except when we can directly solve the dual problem. In practice,one should consider a criterion of the form,

$$|g(\mu^k, \nu^k) - g(\mu^{k-1}, \nu^{k-1})| \leq \varepsilon$$

Dual Ascent Method

The dual ascent method associated to the dual problem consists in updating the dual variables at a given iteration (k) with the feedback of the aggregated optimal local response of the prosumers: the aggregator only has to have access to the sum of prosumer response to shadow prices. Therefore this method allows for privacy as prosumers can be bundled together. The algorithm reads

Algorithm 1 Dual Ascent Method

Initialization: $k = 0$, $\mu := \mu_0 \geq 0$ and $\nu := \nu_0$
while $f^\star - g(\nu^k, \mu^k) \geq \varepsilon$ **do**
  $k + 1 \leftarrow k$
  (1) Find the optimal local solutions EV_i and G_i
  for $i = 1$ to N **do**
    $EV_i^k, G_i^k = \operatorname{argmin} G_i^\top (p - \nu^k) + EV_i^\top B\mu^k + \mathcal{R}_i(EV_i, G_i)$ s. to: $\mathfrak{L}_i$
  end for
  (2) Update the dual variables μ and ν
  $\mu^{k+1} := \left(\mu^k + \alpha c + \alpha \sum_{i=1}^N B^\top EV_i^k\right)_+$

  $\nu^{k+1} := \nu^k - \alpha \frac{1}{\rho} C^{-1} \nu^k - \alpha \sum_{i=1}^N G_i^k$
end while

Where α is the step-size (we will provide an explicit choice of step size in the next paragraphs). Let us derive the convergence rate of this dual ascent algorithm. Let us re-write the Lagrangian

$$g(\nu, \mu) = g_0(\nu, \mu) + \sum_{i=1}^N g_i(\nu, \mu)$$

with

$$g_0(\nu,\mu) = -\frac{1}{2\rho}\nu^\top C^{-1}\nu + c^\top\mu$$
$$g_i(\nu,\mu) = \min_{EV_i,G_i,\ \textbf{s.to.}\ \mathfrak{L}_i} G_i^\top(p-\nu) + EV_i^\top B\mu + \mathcal{R}_i(EV_i,G_i)$$

The gradient of g_0 w.r.t. (ν,μ) is given by

$$\nabla g_0 = \Big[-\frac{1}{\rho}C^{-1}\nu \ ; \quad c\Big] \tag{2.15}$$

Therefore, the Hessian of g_0 satisfies the following bound

$$\frac{1}{\rho\underline{\lambda}}I \succeq -\nabla^2 g_0 \tag{2.16}$$

where $\underline{\lambda}$ is the smallest eigenvalue of C. Using Danskin's Theorem (refer to[26]) to derive the gradient of g_i w.r.t. (ν,μ) yields:

$$\nabla g_i = \Big[-G_i^\star(\nu,\mu) \ ; \quad B^\top EV_i^\star(\nu,\mu)\Big] \tag{2.17}$$

Where

$$EV_i^\star(\nu,\mu), G_i^\star(\nu,\mu) = \underset{s.to.\ \mathfrak{L}_i}{\operatorname{argmin}} G_i^\top(p-\nu^k) + EV_i^\top B\mu^k + \mathcal{R}_i(EV_i,G_i)$$

Note that the local optimization problems are strongly convex and therefore the local solutions are unique for a given $\{\nu,\mu\}$. In the following, we will show that this gradient is smooth. Given (ν_1,μ_1) and (ν_2,μ_2), let $i\in[N]$ and drop the index i temporarily, we denote the corresponding optimal solutions by $G_1^\star.EV_1^\star$ and $G_2^\star, EV_2^\star$ respectively. Considering the L2 regularization (2.7). Using the first order optimality condition for convex constrained problems on the local optimization problems

$$\begin{cases}(p-\nu_1+\delta G_1^\star)^\top(G-G_1^\star)\geq 0,\ \forall G \text{ satisfying the local constraints}\\ (p-\nu_2+\delta G_2^\star)^\top(G-G_2^\star)\geq 0,\ \forall G \text{ satisfying}(\cdots)\end{cases}$$

As $G_1^\star.G_2^\star$ satisfy the local constraints, we have that

$$\begin{cases}(p-\nu_1+\delta G_1^\star)^\top(G_2^\star-G_1^\star)\geq 0\\ (p-\nu_2+\delta G_2^\star)^\top(G_1^\star-G_2^\star)\geq 0\end{cases}$$

Summing these two inequalities gives

$$(\nu_2-\nu_1)^\top(G_2^\star-G_1^\star)\geq\delta\|G_1^\star-G_2^\star\|^2$$

Which gives, using Cauchy Schwartz inequality,

$$\|G^\star(\nu_1,\mu_1) - G^\star(\nu_2,\mu_2)\| \leq \frac{1}{\delta}\|\nu_2 - \nu_1\| \tag{2.18}$$

using the same approach, we can show that

$$\|EV^\star(\nu_1,\mu_1) - EV^\star(\nu_2,\mu_2)\| \leq \frac{\|B\|^2}{\delta}\|\mu_1 - \mu_2\|$$

all in all the above gradient inequalities allow us to prove that g_i is $\frac{1}{\delta}\|B\|^2$-smooth ($\|B\| \geq 1$). For $T \geq 10$ we have $\|B\| \leq T$, therefore we will consider the g_i to be $\frac{T^2}{\delta}$ smooth. Since g_i is the point-wise minimum of a jointly concave function of (ν,μ) it is also a concave function. Using this fact, concavity, (2.16) and (2.17), we conclude that g is a L-smooth differentiable concave function with

$$L = \frac{1}{\rho\underline{\lambda}} + \frac{NT^2}{\delta} \tag{2.19}$$

Remark that from this we have directly $L = O(N)$ (T and $\underline{\lambda}$ are independent of N). We will use this result extensively in our convergence rate analysis in this chapter and chapter 3. From the smoothness of the dual function, we are able to prove the following convergence results,

Theorem 2.4.1. *Let $\mu^\star$ and $\nu^\star$ be optimal variables and consider Algorithm 1. A step-size choice of $\alpha = \frac{1}{L}$ leads to:*

$$f^\star - g(\mu^k,\nu^k) < \frac{M}{2k}(\|\nu_0 - \nu^\star\|^2 + \|\mu_0 - \mu^\star\|^2)$$

Additionally, the oracle complexity (or wosrt case complexity) of Algorithm 1 is:

$$K(\varepsilon) = \frac{L}{2\varepsilon}(\|\nu_0 - \nu^\star\|^2 + \|\mu_0 - \mu^\star\|^2)$$

In other words our algorithm is $O(\frac{N}{k})$

Remark 2.4.1. *A proof of convergence (with derived rate) for L-smooth convex optimization problems can be found in [20]. We include a proof for comprehensiveness.*

Proof. Let $k \geq 1$ and $\omega^k := [\nu^k;\mu^k]$, then by L-smoothness of g:

$$g(\omega^{k+1}) - g(\omega^k) \leq \nabla g(\omega^k)^\top \left(\omega^{k+1} - \omega^k\right) + \frac{L}{2}\|\omega^{k+1} - \omega^k\|^2$$

Using the non-expansiveness property of $(\cdot)_+$ and the Cauchy-Schwartz inequality we get

$$g(\omega^{k+1}) \geq g(\omega^k) + \left(\alpha - \frac{L\alpha^2}{2}\right)\|\nabla g(\omega_k)\|^2$$

The maximum of the quadratic function $t \mapsto t - \frac{Mt^2}{2}$ is achieved at $t = \frac{1}{M}$. Substituting this step-size choice, taking the opposite of the inequality and adding $f^\star = g(\omega^\star)$ ($\omega^\star \in \operatorname{argmax} g(\omega)$) on both sides yields

$$g(\omega^\star) - g(\omega^{k+1}) \leq g(\omega^\star) - g(\omega^k) - \frac{1}{2L}\|\nabla g(\omega_k)\|^2 \tag{2.20}$$

Consequently, the error sequence $\{g(\omega^\star) - g(\omega^{k+1})\}_{k=0}^{+\infty}$ is decreasing in k. By concavity of g we have that

$$g(\omega^\star) \leq g(\omega^k) + \nabla g(\omega^k)^\top \left(\omega^\star - \omega^k\right) \tag{2.21}$$

Combining (2.20) and (2.21) leads to

$$\begin{aligned} g(\omega^\star) - g(\omega^{k+1}) &\leq \nabla g(\omega^k)^\top \left(\omega^\star - \omega^k\right) - \frac{1}{2L}\|\nabla g(\omega^k)\|^2 \\ &= \frac{L}{2}\left(\|w^\star - w^k\|^2 - \|w^\star - w^k - \frac{1}{L}\nabla g(w^k)\|^2\right) \end{aligned}$$

Using the fact that for $x \geq 0$ and for all $y \in \mathbb{R}^n$,

$$\|x - y_+\| \leq \|x - y\|$$

And the dual ascent update, we have that

$$\|w^\star - w^k - \frac{1}{L}\nabla g(w^k)\|^2 \geq \|w^\star - w^{k+1}\|$$

Therefore

$$f^\star - g(w^{k+1}) \leq \frac{L}{2}(\|\omega^k - \omega^\star\|^2 - \|\omega^{k+1} - \omega^\star\|^2)$$

Using the fact that the error is decreasing, it is possible to write this bound as a telescopic sum in the following way:

$$\begin{aligned} g(\omega^\star) - g(w^k) &\leq \frac{1}{k}\sum_{j=0}^{k-1}(g(\omega^\star) - g(w^j)) \\ &\leq \frac{M}{2k}\sum_{j=0}^{k}(\|\omega^j - \omega^\star\|^2 - \|\omega^{j+1} - \omega^\star\|^2) \\ &\leq \frac{M}{2k}\|\omega^0 - \omega^\star\|^2 \end{aligned}$$

Which establishes the claim. □

2.5 Study Case for Convergence Analysis

In this section we illustrate the proposed dual ascent algorithm with a case study in California under CAISO including 100 modeled prosumers with PEV and PV panels. We will only focus on the convergence of the dual algorithm and we will not provide examples of resulting optimal power profiles for PEV charging and market participation – but we will do so in Chapter 4. To begin, we describe the simple price forecast model we use for the day ahead market.

Day-Ahead Energy Market price Model

The day-ahead market in California (CAISO) consist of three separate processes. The first process assess whether the bidders may be able to exert market power, in a second step CAISO forecasts the level of supply needed to meet the demand for each hour. Finally, the additional plants that must be ready to generate electricity are determined. Bids and schedules can be submitted up to seven days in advance, but the market closes at noon the day before and the results are available at 1pm. A bid for a given hour consists in submitting a price and corresponding quantities. Nevertheless, there is the possibility for a power producer to only submit a quantity, in fact, according to CAISO website, about 70% goes through the market as self scheduled or as a price taker. We will take that perspective in the following study case: this is all the more relevant to this case study as we will consider that the aggregator only uses electricity from the grid (i.e. no power injection, $\forall i,\ \underline{G_i} = 0$).

Three years of data (Jan 2013- Dec 2015) have been collected from CAISO across the Pacific Gas and Electricity (PG&E) service territory. These data are publicly available [38]. An online prediction model for DAM prices is built using random forest regression with 10 trees [122]. The features for prediction are:

- hourly demand forecast (provided by CAISO)
- year, month, day, and hour

The model is initially trained with data from 2013, and then recursively updated. Let us denote $\overline{p_D} = \{p_1, ...p_D\}$ the available price history. Let us denote $\mathcal{M}_D$ the random forest regression model corresponding to this historical data. In order to predict price in the next day $\hat{p}_{D+1} \in \mathbb{R}^{24}$, the model $\mathcal{M}_D$ is used. For the next day, an updated model $\mathcal{M}_{D+1}$ using historical data $\overline{p_{D+1}}$ is used to predict the price $\hat{p}_{D+2}$, etc. In other words, we use an online model with a retrospective rolling horizon of D days. The obtained DAM price prediction model has a root mean square error (RMSE) of 3.5 USD and a mean absolute percentage error (MAPE) of 8.4 %.

A Gaussian mixture model (GMM) is then fitted to the ex-post prediction error. Nevertheless, we find that the lowest Bayesian information criterion score is attained for a single GMM

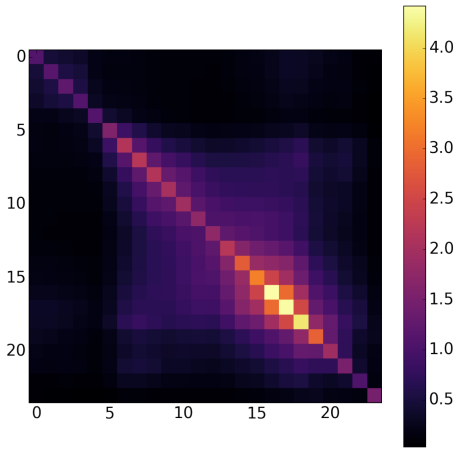

Figure 2.2: Sparse covariance matrix estimation for DAM prices ($/MWh)

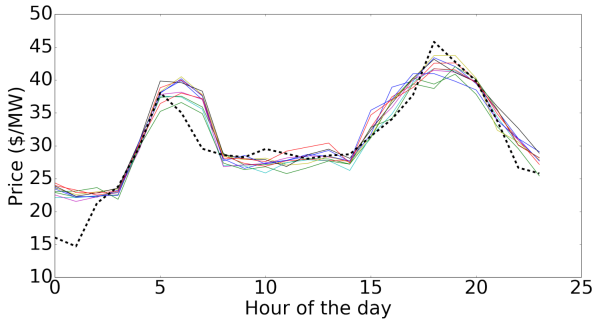

Figure 2.3: Day Ahead Market price predictions

component. This reinforces the choice for using a simple multivariate Gaussian distribution to model price uncertainty (as we assumed in the previous sections). A sparse covariance matrix estimator [122, 108] is then fitted to the ex-post DAM price forecasting error with a L_1 regularization parameter tuned to 5. The sparse covariance matrix consists in solving the following convex problem,

$$\hat{C}^{-1} = \text{argmin}_{\Theta \succ 0} \frac{1}{m} \mathbf{Tr}(\Theta C) - \log(\det(\Theta)) + \gamma \|\Theta\|_1$$

Where C is the empirical covariance matrix

$$C = \frac{1}{D} \sum_{d=1}^{D} (p_d - \hat{p}_d)(p_d - \hat{p}_d)^\top \in \mathbb{S}_+^T$$

Sparse covariance is a log likelihood estimator estimator that allow to delete correlation between hours that are less significant while empirically increasing the covariance matrix condition number (which has a favorable impact on the projected gradient ascent convergence) [108]. In Figure 2.2, we display a heat-map for this covariance matrix: from the data we collected, we can see that the error is correlated mostly during the day and particularly at peak hour around 6 PM (at night only diagonal elements appear). Remark that from our multivariate normal distribution, we can produce more than one prediction for the price using Cholesky factorization [26], as the covariance is positive semi-definite, we can compute its matrix square root, $C = LL^\top$, we have that

$$p = \hat{p} + Lu \sim \mathcal{N}(\hat{p}, C), \text{with} \quad u \sim \mathcal{N}(0, I)$$

which gives a way to simulate more than one price. We illustrate this in Figure 2.3 where the dashed line is the actual price, and the colored lines are different predictions using this method. We do not use more than one prediction in our optimization method, the regularization incorporates already the uncertainty of the price. Using different price predictions would consist in a scenario based approach which we did not take here.

Prosumer Modeling and Parameters

A total of 100 prosumers are modeled. Each prosumer is considered to have a PEV with identical battery size of 24 kWh and itineraries based on National Household Travel Survey [135]. We assume PEVs cannot provide power to the grid: $\underline{G}_i = 0$. As the mobility data [135] does not allow to fit a stochastic model as described in (2.8) (we do not have enough mobility data to build such a statistical model). Therefore, we make the assumptions that

$$\underline{\mathbf{sum}(ev)}_\eta(t) := (1+0.05) \sum_{i\in[N]} \underline{ev}_i(t) \leq A\mathbf{sum}(EV)(t) \leq (1-0.05) \sum_{i\in[N]} \overline{ev}_i(t) := \overline{\mathbf{sum}(ev)}_\eta(t)$$

$$\underline{\mathbf{sum}(EV)}_\eta(t) := (1+0.05) \sum_{i\in[N]} \underline{EV}_i(t) \leq \mathbf{sum}(EV)(t) \leq (1-0.05) \sum_{i\in[N]} \overline{EV}_i(t) := \overline{\mathbf{sum}(EV)}_\eta(t)$$

Table 2.2 details parameter values used for the simulations.

Simulation Results

The projected gradient ascent algorithm described in Section 2.4 is implemented in Matlab with the parameters displayed in Table 2.2. In Figure 2.4, $g(\omega_k)$ is plotted with respect to the number of iterations. The primal optimal cost was obtained by solving the primal problem with CVX [64]. In the proof of Theorem 2.4.1 we show that the choice of $\alpha = 1/L$ with L given by (2.19) guarantees convergence to $f^\star$. Using this theory we have that $\alpha \simeq 2.10^{-7}$,

Table 2.2: Model parameter values for aggregation

N	100
Δt	1 hour
ρ	1
δ	10^{-2}
L_i	Heterogeneous, same load data scaled from [38]
	with uniform independent noise added for each prosumer
S_i	Heterogeneous, same production data generated via [87]
	with uniform independent noise added for each prosumer
η	90 %
$\overline{EV}_i$	1.4 kW $\forall i$
$\underline{ev_i}, \overline{ev_i}$	Heterogeneous, generated from [135]
$\overline{G_i}, \underline{G_i}$	$+/-$ 10 kW $\forall i$
p, C	cf. Section 2.5

but in practice we find that $\alpha = 10^{-5}$ gives better convergence behavior because the derived smoothness value M is conservative. From Theorem 2.4.1 we have that:

$$f^\star - \frac{L}{2k}\|\omega_0 - \omega^\star\|^2 < g(\omega^k)$$

This upper-bound is represented by the solid blue curve in Figure 2.4. For this simulation we take $\frac{1}{M} = \alpha = 10^{-5}$ to illustrate the fact that the derived M-smoothness bound is indeed conservative. We find that our simulation is consistent with theorem 2.4.1:

- convergence is sub-linear because the dual problem is not strongly convex
- the derived upper-bound is verified in practice

From Fig. 2.4, it can be seen that the scheme converges approximately in less than 50 iterations. It practice, this means that the aggregator needs to communicate less than 50 times with all the prosumers in order to reach a consensus. Nevertheless, as the convergence is not linear, extra precision requires more iterations. However, since our main objective is to schedule the aggregated load $\mathbf{sum}(G)$ in the DAM, this algorithm offers an appropriate convergence rate that is robust and theoretically guaranteed.

Remember we have

$$L = \frac{1}{\rho\underline{\lambda}} + \frac{NT^2}{\delta}$$

Therefore $L = O(N)$ which directly gives a $O(N)$ complexity of the dual ascent algorithm with respect to the number of prosumers. This can be problematic if we consider one million prosumers as we could expect convergence in half a million iterations (if we extend the results from our simulation). We will propose a solution to that scalability issue in Chapter 3 with Accelerated and Stochastic methods. It is important to note that the aggregator computational burden would still be small in that case.

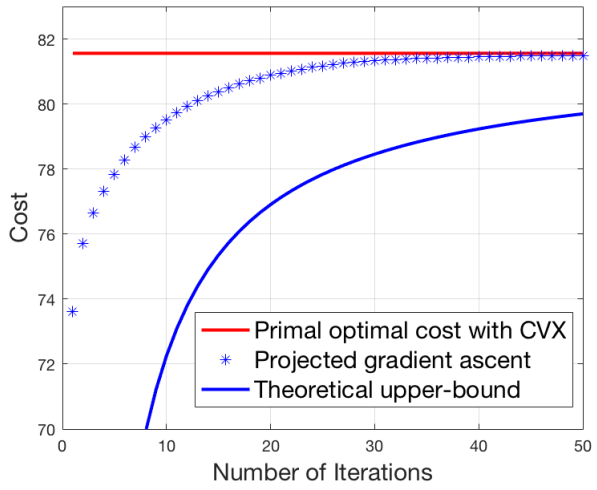

Figure 2.4: Study case: convergence of dual ascent to the optimal primal cost for 100 prosumers

Conclusion and extensions

In this chapter we studied a scheduling optimization algorithm for the aggregation of residential energy prosumers in the day ahead market, taking into account uncertainty in market prices and PEV flexibility. More precisely a model for prosumers with PEVs and PV panels is considered. We considered stochastic constraints on PEV mobility and presented a methodology that can be extended to various sources of electricity production, flexibility and uncertainties.

We saw how the structure of the primal optimization problem can be exploited to distribute the objective among the prosumers. In a distributed scheme, the aggregator broadcasts price signals (i.e. dual variables) to the prosumers, and the prosumers send back their corresponding

energy consumption profile. These price signals are then updated by the aggregator until a consensus is reached.

We have shown that the price signal can be iteratively obtained via dual ascent. Nevertheless, we have pointed out potential limitations of this algorithm for a large number of prosumers. We will provide an answer to this issue in the next Chapter.

Chapter 3

Acceleration and Stochastic Methods for Distributed Energy Resources

In this chapter, we study algorithms extension to the dual ascent method that was proposed in the previous chapter: we notably propose the use of stochastic dual ascent, Nesterov accelerated gradient ascent and a stochastic version of it. We will not prove the convergence rates but instead rely on the existing literature that derive such results. We then discuss the different rates of convergence and their implications, and explain why Stochastic Nesterov Accelerated Gradient Ascent can be seen as a scalable and a practical algorithm.

3.1 Stochastic Gradient

We have seen that for a large number of prosumers, our distributed algorithm might require a proportionally large number of iterations to converge. One engineering idea to tackle this issue, could be to regroup prosumers into clusters: given the power profiles and mobility behavior, we regroup similar prosumers into the same group. If there are M clusters, we could consider that there are M prosumers instead of N, this is particularly interesting when $N >> M$. Even though we believe this approach to be pertinent, proving any convergence or statistical results with this new model compared to the original one is possible. An approach similar to this idea and with provable convergence is stochastic gradient. Stochastic gradient consists in choosing at random a subset M of the prosumers at each dual iteration and use their response to the dual variable to update it. More precisely, we draw uniformly at random at iteration k, a subset $S_k \subset [N]$, the dual *unbiased* dual update is given by

$$\begin{cases} \nu^{k+1} = \nu^k - \frac{\alpha^k}{\rho} C^{-1} \nu^k - \alpha^k \frac{N}{|S^k|} \sum_{i \in S^k} G_i^k \\ \mu^{k+1} = \left(\mu^k + \alpha^k c + \alpha^k \frac{N}{|S^k|} \sum_{i \in S^k} B^\top E V_i^k \right)_+ \end{cases}$$

Where we used the gradient formulas from the previous chapter, (2.15) and (3.1). Remark that these updates are very similar to the updates in algorithm 1. The presence of the term

$N/|S^k|$ ($|\cdot|$ denotes the cardinal of the subset) is here to have an unbiased gradient estimation, meaning that

$$\mathbb{E}\Big(\frac{N}{|S^k|}\sum_{i\in S^k} G_i^k\Big) = \sum_{i=1}^{N} G_i^k$$

Remark that the step size is not consider to be constant. Instead we will take $\alpha^k = \frac{\alpha_0}{\sqrt{k}}$ where α_0 is a hyperparameter that can be tuned by the aggregator. For such a choice of step-size, if α_0 is chosen small enough we have that the worst convergence rate to the optimal variable is $O(\frac{N}{\sqrt{k}})$ (We use results on the convergence of stochastic convex methods from [115]). Remark that in practice, we will only use a constant step size. This guarantee of convergence relies on the fact that the dual variables are bounded and the fact that the estimate of the gradient is bounded by

$$\|\frac{N}{|S^k|}\sum_{i\in S^k} G_i^k\| \leq N\sum_{i=1}^{N}\overline{G}_i$$

The rate of convergence is slower with respect to the number of iterations, nevertheless computational burden per iteration is reduced. Nevertheless the convergence rate is still linear in the number of prosumers. This is of course a worst case rate, in practice - refer to the study case - we will see that the convergence is similar to that of the non - stochastic methods. This result could point towards the fact that our engineering solution consisting in clustering prosumers is more advantageous. Nevertheless, if we assume that the variance is low between consumption and mobility profiles, the convergence of this algorithm similar to the one that would be obtained with the clustering approach. For example, if we assume that there is only one profile of prosumer with no variance, than choosing one prosumer at random is similar to modeling all prosumers as a single individual. This is not exactly true if there are K clusters to be considered (with no or very low variance in each cluster), but a stochastic algorithm that picks at random individuals within each cluster would be equivalent to the clustering approach. In other words, our stochastic algorithm can be improved given a more sophisticated statistical model for the prosumers: in this section we only assumed the prosumers' profiles to be i.i.d.

In the next section we will show that we can reduce the rate of convergence to be proportional to $\sqrt{N}$ - compared to N - via a gradient acceleration method.

3.2 Accelerated Gradient

By construction the gradient ascent method guarantees the increase of the dual function $g(w)$ at each iteration, which might be short-sighted. Accelerated methods, exploit the information from past iterates and adds momentum to the iterates w^k. There are various forms of gradient acceleration methods such as the Heavy-Ball method [123]. In this chapter, we focus our attention on a particular form of gradient acceleration: Nesterov acceleration.

The corresponding dual update is given by

$$\begin{cases} \nu^{k+1} = \nu^k + \frac{k-1}{k+2}(\nu^k - \nu^{k-1}) + \alpha \nabla g_\nu(w) \\ \mu^{k+1} = \left(\mu^k + \frac{k-1}{k+2}(\mu^k - \mu^{k-1}) + \alpha \nabla g_\mu(w)\right)_+ \end{cases}$$

This form of update is in between the classic gradient update and extrapolation, with each iteration having the same cost as gradient updates. Moreover we have from [114] (or alternatively [17, 118, 61, 113, 16, 146]) that

$$f^\star - g(w^k) \leq \frac{2L\|w_0 - w^\star\|}{k^2}$$

Interestingly the chosen ratio $\frac{k-1}{k+2}$ and the use of only one historical value for w is optimal for the convergence rate (meaning that this is the best result - even using more of the history w - that can be theoretically achieved using momentum). From the previous inequality, we have that the algorithm worst case convergence is proportional to $O(\frac{\sqrt{N}}{k^2})$. There are two benefits when we compare this to dual ascent: first, although sub-linear, the convergence is $O(\frac{1}{k^2})$ instead of $O(\frac{1}{k})$, second the algorithm scales better with respect to the number of prosumer: if convergence for 100 prosumers is achieved for 10 iterations, we can expect convergence for 10^6 prosumers in less than 1,000 iterations (we would expect 100,000 for dual ascent). Even though there is no theoretical result for stochastic Nesterov accelerated gradient we are aware of, we could consider the following updates

$$\begin{cases} \nu^{k+1} = \nu^k + \frac{k-1}{k+2}(\nu^k - \nu^{k-1}) - \frac{\alpha^k}{\rho} C^{-1}\nu^k - \alpha^k \sum_{i \in S^k} \frac{N}{|S^k|} G_i^k \\ \mu^{k+1} = \left(\mu^k + \frac{k-1}{k+2}(\mu^k - \mu^{k-1}) + \alpha^k c + \alpha^k \frac{N}{|S^k|} \sum_{i \in S_k} B^\top EV_i^k\right)_+ \end{cases}$$

Where similarly to the previous section, we draw uniformly at random at iteration k, a subset $S_k \subset [N]$. We will see that in practice (study case) this algorithm performs well for the task of coordinating the prosumers to the optimum.

3.3 Linear convergence for a large number of prosumers

The dual ascent guaranteed rate of convergence being sub-linear (i.e. $O(\frac{1}{k})$) stems from the fact that we consider the robust mobility constraints. Indeed, without them, the dual function would only be a function of ν and

$$g(\nu) = -\frac{1}{2\rho}\nu^\top C^{-1}\nu + c^\top \mu + \sum_{i=1}^{N} \min_{EV_i, G_i, \text{ s.to. } \mathfrak{L}_i} G_i^\top (p - \nu) + \mathcal{R}_i(EV_i, G_i)$$

Using Danskin's Theorem [26] to derive the gradient yields

$$\nabla g_\nu = \frac{1}{\rho} C^{-1}\nu - \sum_{i=1}^{N} G_i^\star(\nu) \tag{3.1}$$

Where $G_i^\star(\nu)$ is the unique solution to the local optimization problem at $i \in [N]$ given ν. Similarly to inequality (2.18), we have that

$$\|G_i^\star(\nu_1) - G_i^\star(\nu_2)\| \leq \frac{1}{\delta}\|\nu_1 - \nu_2\|$$

Therefore $g(\cdot)$ is $L := \frac{1}{\rho}\underline{\lambda} + \frac{N}{\delta}$-smooth concave function. We also have, using the spectral theorem

$$C^{-1} \succeq \frac{1}{\bar{\lambda}} I$$

Which proves that g is $l := \frac{1}{\rho\bar{\lambda}}$-strongly concave. Using this structure, we can show in a few steps that the convergence of the dual ascent algorithm with

$$\nu^{k+1} = \nu^k + \frac{1}{L}\nabla g(\nu)$$

is linear, meaning that the convergence to the optimal dual variable is $O(r^k)$ with $0 < r < 1$. Indeed,

$$\begin{aligned}\|\nu^{k+1} - \nu^\star\|^2 &= \|\nu^k - \nu^\star + \frac{1}{L}\nabla g(\nu^k)\|^2 \\ &= \|\nu^k - \nu^\star\|^2 + \frac{1}{L}\nabla g(\nu^k)^\top(\nu^k - \nu^\star) + \frac{1}{L^2}\|\nabla g(\nu^k)\|^2\end{aligned}$$

From strong concavity we have the inequality - refer to (1) in Notation & Background

$$\nabla g(\nu^k)^\top(\nu^k - \nu^\star) \leq (g(\nu^k) - g(\nu^\star)) - \frac{l}{2}\|\nu^k - \nu^\star\|^2$$

Therefore,

$$\|\nu^{k+1} - \nu^\star\|^2 \leq (1 - \frac{l}{L})\|\nu^k - \nu^\star\|^2 - \frac{2}{L}(g(\nu^k) - g(\nu^\star)) + \frac{1}{L^2}\|\nabla g(\nu^k)\|^2$$

From smoothness of g, concavity and optimality of $\nu^\star$, we also have that

$$g(\nu^k) - g(\nu^\star) \leq -\frac{1}{2L}\|\nabla g(\nu^k)\|^2$$

Hence

$$\|\nu^{k+1} - \nu^k\|^2 \leq (1 - \frac{l}{L})\|\nu^{k+1} - \nu^\star\|^2$$

Therefore the sequence $\{\|\nu^\star - \nu^k\|^2\}$ is a $r = 1 - \frac{l}{L}$ contraction. To guarantee that $\|\nu^k - \nu^\star\| \leq \varepsilon$, we have a worst case number of iterations needed (oracle complexity) of

$$K(\varepsilon) = 2\frac{\log(\frac{\varepsilon}{\|\nu^0 - \nu^\star\|})}{\log(1 - \frac{l}{L})}$$

As r approaches one, the algorithm becomes slower. In fact if $r = 1$ this bound cannot be used. We need to use the theorem 2.4.1 from the previous chapter, and we have a sub-linear convergence. Note that l does not depend on the number of prosumers, as it is related to the covariance matrix of DAM price prediction, therefore it is easy to show that as $N \to \infty$, $r \to 1$. In practice for N large the convergence is still sub-linear, and the analysis we had in the previous chapter is not affected. This also means that the Nesterov accelerated method we proposed in the previous section still keeps its advantage (Remark that in the strongly concave case, the acceleration provides faster convergence with a dependence on N that is also $\sqrt{N}$). For simplicity of implementation, we will therefore consider in the study case the DAM optimization problem without the inequality global inequality constraints.

Simulation Results

Compared to the last section, we choose a slightly higher number of prosumers and smaller hyperparameter values for DAM risk and L2 regularization. We report the values that are not the same as the previous chapter in the following table,

Table 3.1: Some parameter Values for the Model

N	120
ρ	10^{-2}
δ	10^{-2}
α	$3 \cdot 10^{-6}$

Numerically, we have that $\frac{1}{L} \simeq 1.9 * 10^{-6}$, in practice the algorithm diverges for a step size $\rho > 4.10^{-6}$, therefore we made the choice of step size $\alpha = 3 \cdot 10^{-6}$. From figure 1 and 2, it can be seen that the algorithms for gradient ascent and accelerated gradient ascent behave as predicted. The upper bound for a precision of 10^{-3} gives an oracle complexity of 42 iterations, which is observed here in practice. We also remark that the accelerated method converges in approximately 10 iterations. We can also see that the stochastic versions of both algorithm both follow the non-stochastic one.

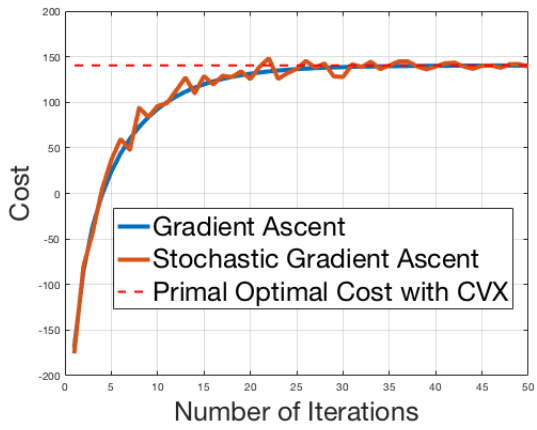

Figure 3.1: Gradient Ascent and Stochastic Gradient Ascent

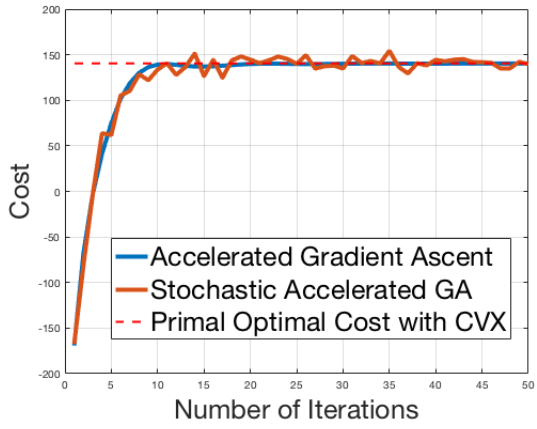

Figure 3.2: Accelerated Gradient Ascent and Stochastic Accelerated Gradient Ascent

Conclusion

In this chapter we have seen how we can improve the distributed methods of Chapter 1 by tackling the following limitations: (i) scaling with respect to the number of prosumers, (ii) removing the need for each prosumer to compute their solution for each dual variable iteration.

Chapter 4

Convex Optimization Model for RTM Participation

In the previous chapter we briefly described the functioning of the DAM. Day Ahead Markets, when they exist, have a common way of functioning in many countries. Real time markets and ancillary services are more specific: the way they operate vary location from location. In this Chapter we will focus on the RTM in CAISO. We will briefly mention the functioning of RTM in other countries, and we believe that the methodology and convex optimization models we propose can be extended across different markets. The notation for this chapter are given in table 2.1.

Disclaimer

We will see that the RTM price prediction seems to be the bottleneck for achieving better resultst. Nevertheless it is important to keep in mind that our main objective is to expose optimization models and methods to tackle such problems. We do not claim to use state of the art time series predictions for DAM and RTM prices.

4.1 Real Time Market in CAISO

The real-time market in CAISO has multiple market instruments in which generators can participate. There are three types of generators: (1) internal generators, ones that operate inside of the CAISO balancing area (BA) authority; (2) import and export generators, those that produce power in the BA but for use outside of the BA or those that produce power outside the BA but for use inside the BA; and, (3) dynamic resources, those generators located outside the BA that have telemetry and controls for power delivery inside the BA.

We provide a graphical explanation of the RTM in Figure 4.1. The two time periods that are of most relevance are the 15-minute market and the 5-minute market. Supply-side bids are submitted, at the latest, 75 minutes prior to the start Trading Hour (T-75). For each

hour there are four bids submitted, one for each 15-minute period. For generators that are able to participate in the 5-minute market (i.e. generators internal to the BA and dynamic resources)[1], the bids that are submitted for the 15-minute period will apply to the 5-minute periods within that same time period. So a generator will provide four bids per hour and there will be three prices for each bid, thus 12 prices per hour.

The two real-time markets (i.e. 15-minute and the 5-minute) operate with the use of separate optimizations. The market prices are published 45 minutes and 22.5 minutes prior to close of the 15-minute and 5-minute markets, respectively. Those prices are published on a rolling basis, so for each 15 or 5 minute period it is 45 minutes and 22.5 minutes prior to the start of that trading period.

We denote p_{RT} the prices in the RTM (see 2.1). It is important to note that real time markets can be positive and negative, but on average, we should expect: $\mathbb{E}(p_{RT}) = p_{DA}$ (p_{DA} denotes the prices in the DAM). Nevertheless, in practice we observe, $\mathbb{E}(p_{RT}) \sim 0.93 p_{DA}$. Based on conversations with CAISO management, this is explained by the fact that some renewable generators schedule their production in real time which has a tendency to lower prices (because they have zero marginal cost). Since not all resources participate in both the DAM and RTM, the price difference is not surprising especially considering that many of the low cost renewable resources are self scheduled and participate only in the RTM. Nevertheless, while empirically there exists a price difference between the two markets, we consider the following two conditions for any given time period:

- $p_{RT} > p_{DA} \Rightarrow$ electricity **System is Short**: day ahead supplied schedule is lower than real time demand.
- $p_{RT} < p_{DA} \Rightarrow$ electricity **System is Long**: too much supply was scheduled compared to real time demand

The prices are received through a program called the California Market Results Interface. Figure 4.1 displays the Real Time Market timeline. Two hours are depicted (hour h and hour h+1) for illustrative purposes to indicate that the the bids are rolling. For both hours, the day-ahead market bids are shown with dotted lines and labeled as "DAM Bid". The real-time (15-minute) bids are submitted at T-75 and the results are published at T-45. The bids that were accepted are then depicted with solid lines. The difference between the DAM and RTM is also shown.

[1]With the introduction of the Energy Imbalance Market (EIM), there are now generators that participate in the real-time market that are located outside of CAISO, i.e. type 2 generators and type 3 generators. The majority of these generators can only submit bids for the 15-minute market (type 2 generators), but a subset also participates in the 5-minute market (type 3 generators).

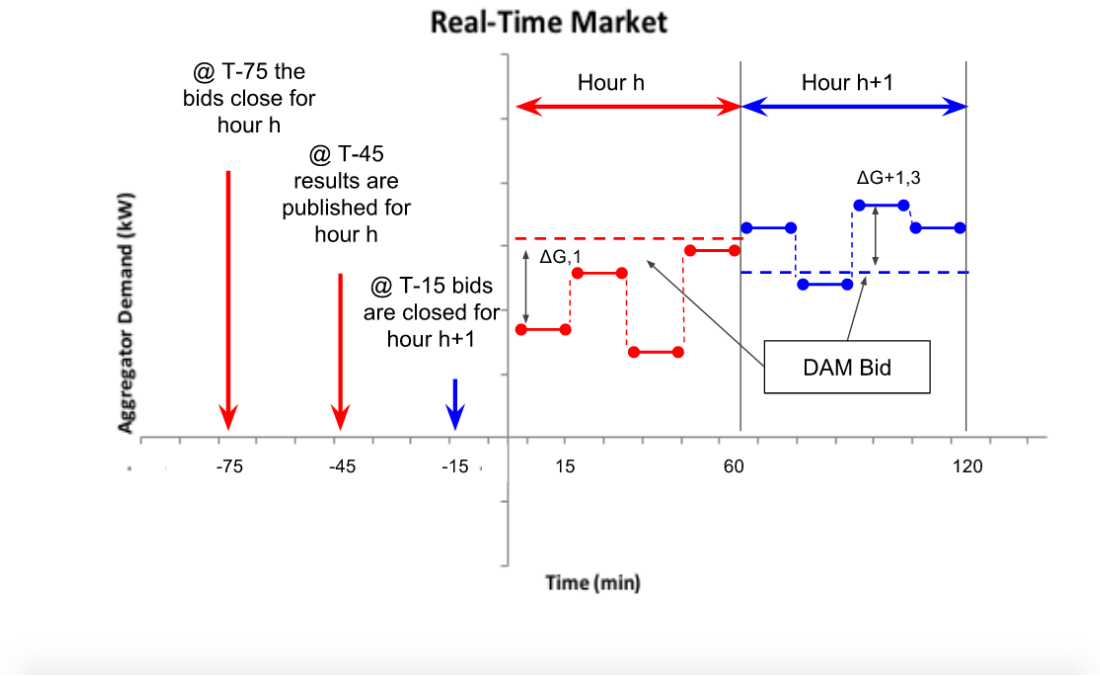

Figure 4.1: Graphical Depiction of the Real Time Market Process

Only the 15 min market interval will be considered in this chapter. Note that the Smart Meters in California allow to give information every 30 min. Therefore, it is already uncertain how the settlement will be made with CAISO.

Day Ahead Market and Real Time Market

The aggregator could increase their revenue by predicting what is going to happen in the real time market while participating in the DAM. Such a strategy consists in betting on the fact that the CAISO forecast of CAISO demand is either short or long for a given hour. Nevertheless we argue that in the CAISO market, this aspect is considered with Virtual Bidding. Virtual bids must be virtual, i.e. not associated with physical bids, and the practice of using physical assets to make virtual bids is forbidden by CAISO. Moreover, market power exertion is assessed by CAISO – whether intentional or not – through a Market Power Mitigation (MPM) process. In the case that bids are deemed to be an exertion of market power, then those bids are rejected and the optimization is run without those bids. Therefore, we argue that the DAM and the real time market optimization should, or must, be independent: the real time market must take the results of the DAM as an exogenous input.

Price prediction and estimation of covariance matrix

As shown in Figure 4.2, the price fluctuation of the RTM (right figure) is much larger than that of the DAM (left figure). During the considered time period, the DAM price was always between 0 USD/MWh and 100 USD/MWh. In contrast, the RTM price spikes relatively frequently and went up as far as 1,000 USD/MWh and down as far as -150 USD/MWh.

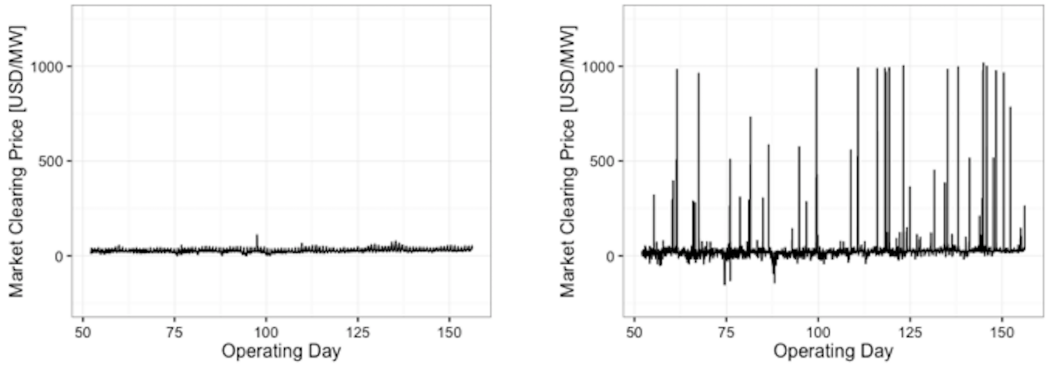

Figure 4.2: Comparison of price fluctuations between the Day-ahead Market (Left) and the Real-time Market (Right)

In the RTM, generation capacity tends to be constrained and supply curve (Short-term Marginal Cost: STMC) of electricity becomes more vertical (more inelastic). As a result, a small change in demand can cause price spikes in either a positive or a negative direction. Such situations are shown in Figure 4.3. The sub-figure on the left shows a change where the supply is relatively elastic and thus the price difference between p_{RT} and p_{DA} is small. In contrast, the figure on the right, the demand moves from the elastic portion of the supply curve to the inelastic portion of the supply curve and the price difference between the DAM and RTM spikes.

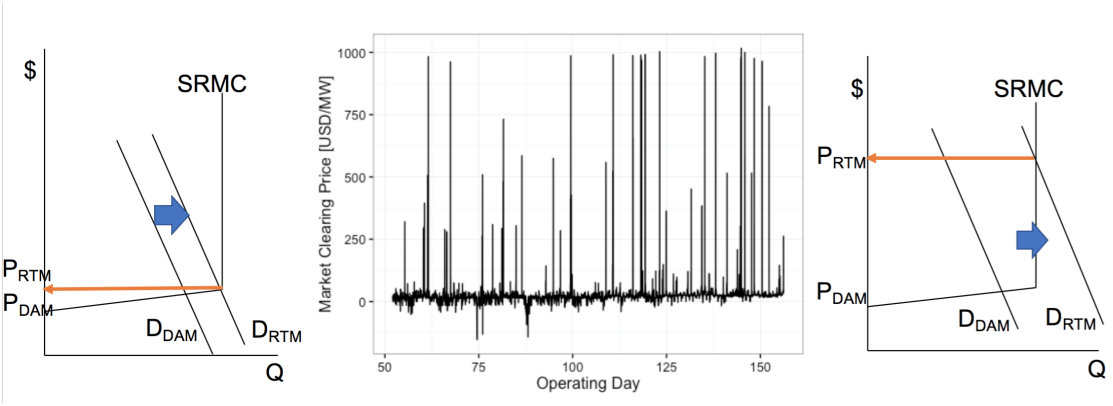

Figure 4.3: Mechanism of price fluctuation in the RTM

This price fluctuation of the RTM price poses a challenge when it comes to price prediction. As in the previous chapter, in our convex optimization model, we will not only minimize the cost in the RTM but also incorporate the uncertainty of the price prediction in the objective function. We will also estimate the covariance matrix that represents the uncertainty of the prediction. We use random forest regression to predict both the DAM and RTM prices. For the DAM price prediction, we used year, date, and hour as features (regressors). In addition to these features for the real-time forecast we use, DAM demand forecast, DAM prices, and RTM demand forecast, as well as operating interval labels (0 min, 15 min, 30 min, and 45 min).

Similar to the previous chapter, we assume multivariate normal distribution around the expected price both for RTM and DAM price prediction:

$$p_{DA} \sim N(\hat{p_{DA}}, C_{DA})$$
$$p_{RT} \sim N(\hat{p_{RT}}, C_{RT})$$

It revealed to be difficult to produce an online prediction model for RTM prices, therefore we kept a *static* one for RTM prediction. For coherence we did the same for DAM price prediction. In our simulations, we get a RMSE of 2$ for DAM and a 15$ RMSE for the RTM. Given the electricity price data prediction error, we use maximum likelihood estimation to estimate the covariance matrices C_{DA} and C_{RT} (we give a heatmap representation of these matrices in the figures 4.1 and 4.1 below). We will use these matrices to incorporate risk management using the Markovitz Portfolio Optimization as in chapter 2.

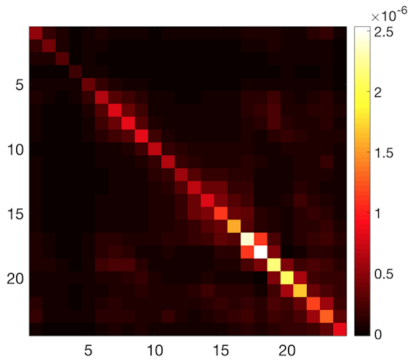

Figure 4.4: Covariance matrix heatmap for the day ahead market

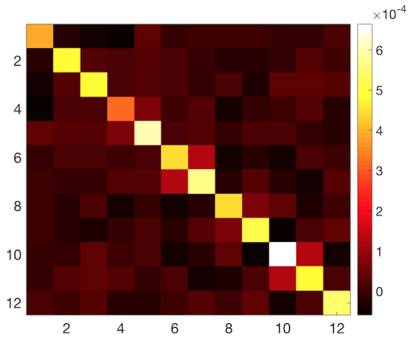

Figure 4.5: Covariance matrix heatmpap for the real time market

4.2 Day-Ahead Market and Real-Time Market Optimization Model

Optimization objective

Let f_{DA} and f_{RT} denote the DA and RT objectives respectively. For clarity, we will use orange color to denote the optimization variables of the RTM. The choice of the grid import/export G (as scheduled in the DAM) or ΔG (as scheduled in the RTM) can be considered as a portfolio problem where the assets is the flexibility, the returns are the DAM prices, and the budget constraint is the flexibility constraints [26]. The choice of the portfolio G involves a trade-off between the expected price and the corresponding variance. For the Day-Ahead, remember that we chose

$$f_{DA} = \underbrace{p_{DA}^{\top} G}_{\text{DA total cost}} + \overbrace{\frac{\rho_{DA}}{2} G^{\top} C_{DA} G}^{\text{DA Risk}}$$

Let us denote the time window $(h : h + 4T_H - 1)$ as $\mathcal{H}$. Similarly, for the RT at hour h we consider the convex objective function

$$f_{RT}(h) = \underbrace{p_{RT}(\mathcal{H})^{\top} \Delta G(\mathcal{H})}_{\text{RT total cost}} + \overbrace{\frac{\rho_{RT}}{2} \Delta G(\mathcal{H})^{\top} C_{RT} \Delta G(\mathcal{H})}^{\text{RT Risk}} \tag{4.1}$$

Let us take a concrete example to understand why (4.1) is the right objective function and understands what happens with negative prices. These examples are given in the following table 4.1.

Table 4.1: Real time market cost function explained

	$\Delta G = +1MW \geq 0$	$\Delta G = -1MW \leq 0$
$p_{RT} = 40 > p_{DA} = 30\$/MW$ (SYSTEM SHORT)	ΔG increases the strain on the system. The aggregator has to pay for this extra quantity at RT market price: 40 \$/MW. Had this extra demand been scheduled in the DA, the cost would have been 10 \$/MW lower	ΔG decreases the strain on the system. The aggregator is paid as if it was providing 1MW of power supply: \$40. Had this demand reduction been included in the DA, the aggregator would have had a cost \$10 higher
$p_{RT} = 20 < p_{DA} = 30\$/MW$ (SYSTEM LONG)	ΔG decreases the strain on the system. The aggregator has to pay for this extra quantity at RT market price: 20 \$/MW. Had this extra demand been scheduled in the DA, the cost would have been 10 \$/MW higher	ΔG increases the strain on the system. The aggregator is paid as if it was providing 1MW of power supply: \$20. Had this demand reduction been included in the DA, the aggregator would have had a \$10 lower cost
$p_{RT} = -10 < p_{DA} = 30\$/MW$ (SYSTEM ULTRA LONG)	ΔG decreases the strain on the system. The aggregator is paid for extra quantity at RT market price: -10 \$/MW! Had this extra demand been scheduled in the DA, the cost would have been 40 \$/MW higher	ΔG increases the strain on the system. The aggregator has to pay as if was providing 1MW of power supply: \$10. Had this demand reduction been included in the DA, the aggregator would have had a cost \$40 lower!

Possible extension for other RTMs

In other countries, the cost function (risk not included) might not be linear. In Germany and the UK, the aggregator would have to face constant imbalance prices. Let δ_+ be the positive imbalance price and δ_- be the negative imbalance price: the aggregator is paid δ_+ for beneficial deviations (e.g. system long and $\Delta G \geq 0$) and has to pay δ_- for deviations that increases strain on the system (e.g. system long and $\Delta G \leq 0$). In the UK, the Imbalance market is symmetric (i.e. $\delta_- = \delta_+$), whereas in Germany $\delta_+ < \delta_-$. Let us introduce this specificities in our Real-Time objective function for which there are four cases to consider, for each the imbalance cost at time t is given by,

1) System Short and $\Delta G \geq 0$:

$$\delta_-(\Delta G(t))_+$$

2) System Short and $\Delta G \leq 0$:

$$-\delta_+(-\Delta G(t))_+$$

3) System Long and $\Delta G \geq 0$

$$\delta_+(\Delta G(t))_+$$

4) System Long and $\Delta G \leq 0$

$$-\delta_-(-\Delta G(t))_+$$

Denoting $\text{short}(t) \in \{0,1\}$ and $\text{long}(t) \in \{0,1\}$, the resulting cost function would be

$$\delta_-\Big(\text{short}^\top(\Delta G)_+ - \text{long}^\top(-\Delta G)_+\Big) + \delta_+\Big(\text{long}^\top(\Delta G)_+ - \text{short}^\top(-\Delta G)_+\Big)$$

This cost function is also convex in ΔG.

Local Model for Prosumers

Here we only display the local prosumer model for the real time model. The DAM prosumer model was thoroughly described in chapter 2. Let $i \in [N]$. We denote $EV_i^\star$ and $G_i^\star$ the optimal schedules from DA. In the real time market optimization, these variables are considered as exogenous inputs as mentioned in the previous section. As there is a time step difference between DAM and RTM that is a multiple of 4, we actually extend the vectors $EV_i^\star$ and $G_i^\star$ the following way,

$$EV_i^\star = [EV_i^\star(1); EV_i^\star(1); EV_i^\star(1); EV_i^\star(1), EV_i^\star(2); EV_i^\star(2); EV_i^\star(2); EV_i^\star(2); \cdots] \in \mathbb{R}^{4T}$$

we do the same with $G_i^\star$

Local Power Balance

The power balance (4.2) establishes the link between the control variables ΔEV_i and ΔG_i as in (2.1)

$$L_i + EV_i^\star + \Delta EV_i \leq S_i + G_i^\star + \Delta G_i \tag{4.2}$$

Local Grid Constraints

At any given node in the distribution network, there is a limit on power import or export as in (2.2)

$$\underline{G}_i \leq G_i^\star + \Delta G_i \leq \overline{G}_i \tag{4.3}$$

Local PEV constraints

The PEV dynamics and state of energy constraints can be summarized as follow as in (2.4)

$$\underline{ev}_i \leq \Delta t \Sigma (EV_i^\star + \Delta EV_i) \leq \overline{ev}_i \tag{4.4}$$

With Σ, trigonal inferior matrix with ones – the integration matrix as described in chapter 2. Finally, the PEVs charging power constraint is given by

$$\underline{EV}_i \leq EV_i^\star + \Delta EV_i \leq \overline{EV}_i \tag{4.5}$$

Note that when the PEV is un-plugged at a given time $t \in [T]$, then $\underline{EV}_i(t) = \overline{EV}_i(t) = 0$ is enforced. For the purpose of conciseness, the set of local constraints (4.2), (4.3), (4.4) and (4.5) are hereby referred to as $\mathfrak{L}_i$.

Model Predictive Control Scheme for Real-Time Operation

We use MPC because of the dynamic aspect of the RTM we described. It is also difficult to predict the price many hours in advance. It also helps reducing the size of the optimization problem at hand. We will us a three hour rolling hours horizon in our simulations. The following algorithm summarizes our MPC strategy for participation in the RTM

Algorithm 2 Model Predictive Control

1: **for** h from 1 to T_H do: **do**

$$\Delta EV^\star(\mathcal{H}, \Delta G^\star(\mathcal{H})) = \underset{\text{s.to.}\mathfrak{L}_i}{\operatorname{argmin}} f_{RT}(\Delta EV(\mathcal{H}, \Delta G(\mathcal{H})), \Delta EV_i(\mathcal{H}), \Delta G_i(\mathcal{H}))$$

Only implement the decision for the first hour: $\Delta EV^\star = \Delta EV^\star(1:4)$, $\Delta G^\star = \Delta G^\star(1:4)$, etc.

2: **End For**

4.3 Simulation and Results

For price prediction and covariance matrix estimation, we collected CAISO's DAM (hourly) and RTM (15 minutes) price and demand forecast data in PG&E area from January 2013 to March 2017.

We used mobility data from 2,000 full electric vehicle's in the bay area. A single solar PV generation data was generated using PVsim (SunPower). The load was modeled taking aggregate load data from CAISO in the PG&E region and adding random noise to it to create heterogeneity between prosumers (idem for prosumer PV production). Most of the data was concatenated in the same CVS file using R. Here we display a one day simulation to observe the behavior of our scheduling and real time operation method. We made a simulation for 100 prosumers, the parameters that were chosen for this simulation are provided in the following table 4.2

Table 4.2: Model parameter values for aggregation in the real time market

N	100
T_H	3 hours
$\overline{G}$	10 kW
ρ_{DA}, ρ_{RT}	1

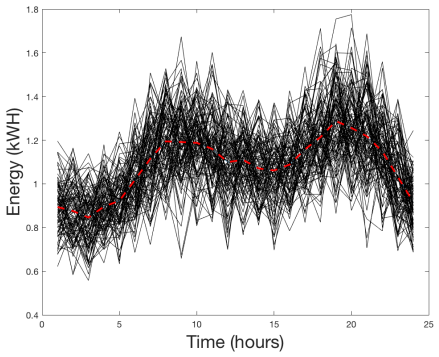

Figure 4.6: Local uncontrollable load consumptions (black curves), Average (red dashed curve)

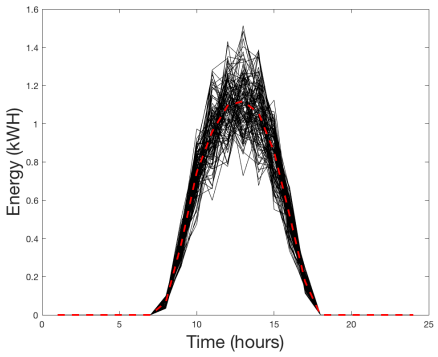

Figure 4.7: Local solar PV production (black curves), Average solar PV production (red dashed curve)

First, figures 4.6 and 4.7 provide a visualization of the average uncontrollable load and PV production as well as the heterogeneity inside the pool for DA (i.e. 1 hour time step). Figure 4.8 provides a visualization of the difference between the DA and RT aggregated

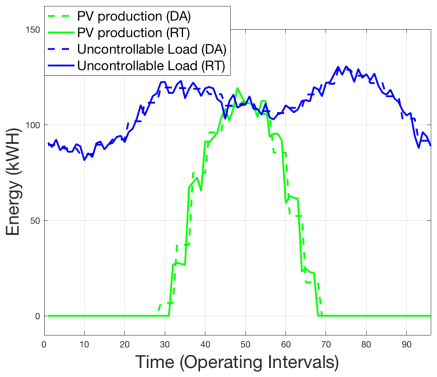

Figure 4.8: Aggregated uncontrollable load PV production

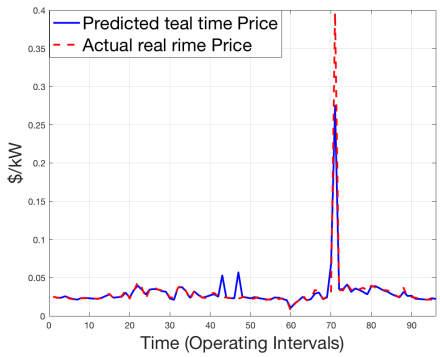

Figure 4.9: Real-Time prices for one day of the simulation

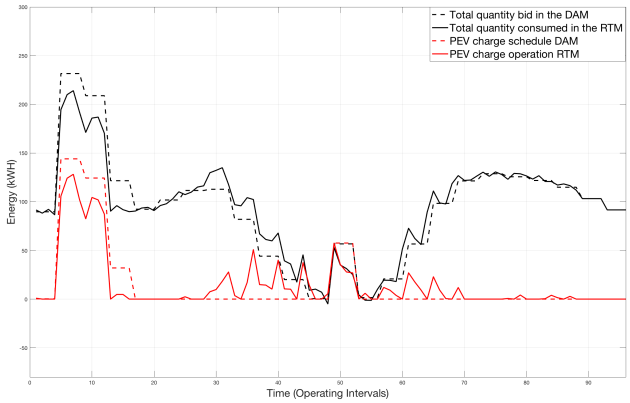

Figure 4.10: DA Schedule and RT operation results: $G^\star$, black dashed curve, $\Delta G^\star$ solid black dashed curve, $EV^\star$ red dashed curve and $\Delta EV^\star$ solid red curve

load an PV production. Figure 4.9 shows the real time price for the day. The results of the MPC convex optimization approach are provided in figure 4.10. This figure shows the difference between the DA schedule and real time operation. In the DA we predict that our clearing price will be \$68.7 for the next day. In reality, when the DA market clears, the aggregator realizes that its cost will be \$71.1. In the RTM, the total supplementary cost over the day is \$0.30. This means that the aggregator is not able to leverage any value in the RT market. Nevertheless, our scheme gives us a way to do real time operation over our pool. It is difficult to determine what costs the aggregator would incur for its real time operation without this scheme, moreover price spikes only appears for 15 min during the chosen day and are generally rare events. Based on our simulations, the aggregator does not take advantage

of this price surge. This is due to the fact that our MPC horizon is too small and there is no available flexibility by the time we reach spike time (around operating interval 70). When the risk aversion parameter for Real Time ρ_{RT} is set to zero, the total cost is -$0.90, which means the aggregator leveraged the RTM to reduce its cost.

4.4 Conclusion

In this chapter we developed a forecast model for real-time markets using random forest regression technique and evaluated a covariance matrix associated to our model. We then proposed a two-step method for managing a pool of prosumers with local production and flexibility. In a first step, the aggregator schedules its total supply and demand curve to CAISO in the day ahead market; in a second step, the aggregator operates in real time using a Model Predictive Control Technique and makes a decision for the optimal deviation from the DAM schedule. To do so, we proposed a convex optimization method similar in structure to that of the DAM formulation of the previous chapters: we can therefore use the same methodology to distribute the problem in real time among prosumers.

Part II

Chapter 5

Hopfield Methods for Combinatorial Optimization

In the previous chapters we showed how we could use distributed methods for tackling DER electricity market integration problems. More precisely, we modeled prosumers with PEVs and PV: remark that we modeled the charging power for PEV as continuous variables, i.e. variables that can be chosen between a minimum power $\underline{EV}$ and a maximum power $\overline{EV}$. In practice the choice for charging powers of PEVs are discrete: only a given set of choices are possible depending on the manufacture of the vehicle, the simplest one being the two following choices: charge (1) or no charge (0). Paradoxically, it is known that this apparent simplicity translates into difficult optimization problems. In the literature about DER integration it is rare to see local flexibility being modeled as hybrid systems (e.g. where some variables are combinatorial and others continuous). In the particular case of DAM market scheduling, we do not believe this is an issue, as this is a schedule, not a real time operation. Even for the RTM application we presented in chapter 4, the possibility of distributing the problem - even though, in that case, there is no proof of convergence for dual ascent - could allow to surpass this issue. This is not the case if the problem cannot be distributed: for instance if we incorporate a model of the grid that connects prosumers. Alternatively, the local problem itself could be too hard to solve. For instance, in the context of smart home, many more decisions for turning on or off device could have to be made (e.g. heating, AC, pool pumps, refrigerators, etc.). Exact solvers could be used in certain cases but as the computational power needed is exponential with respect to the number of decision variables, the use of these solvers becomes not tractable. As discussed later in the chapter, these type of combinatorial problems appear across many applications in energy systems, but also in different fields. In this chapter, we propose a heuristic method that can be used for some of these applications.

More precisely, we present novel first-order heuristic methods that aim at finding candidate solutions to large scale nonlinear combinatorial problems. Such problems are generally NP-hard, i.e. there is no algorithm of polynomial complexity available to compute the optimal solution. Our heuristic methods build on Lagrangian relaxation and numerical versions of Hopfield Neural Networks that we call *Hopfield Methods*. We explain the heuristic behind

Hopfield Neural Networks and how they can be interpreted as mirror descent and then generalize using ideas similar to conditional gradient methods (a.k.a Frank-Wolfe). Finally we prove convergence of these methods as well as a worst case rate of convergence of $\mathcal{O}\left(\frac{1}{k}\right)$ for convex functions.

5.1 Introduction

This chapter addresses the problem of efficient algorithms to recover candidate solutions to large-scale combinatorial optimization problems using a novel first-order heuristic method, which we refer to as *Hopfield Methods*.

Notations

We denote by $x \in \mathbb{R}^p$ the *optimization variable* and by I_p the set $\{1, \cdots, p\}$. We will consider the optimization variable to be restricted to a box $lb \leq x \leq ub$, where the inequalities are interpreted component-wise. Via rescaling, and therefore without loss of generality, we will consider that $x \in [0,1]^p$, we use the notation $(0,1)$ to denote the interior of $[0,1]$. We further denote $I_b \subset I_p$ the index set referring to *binary constrained* variables, such that $|I_b| = b \leq p$ and $b > 0$. We denote the mixed-binary set compactly as

$$\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i \quad \text{where } \mathcal{X}_i = \{0,1\} \text{ if } i \in I_b,\ \mathcal{X}_i = [0,1] \text{ otherwise} \tag{5.1}$$

where operator $\prod$ above is interpreted as the Cartesian product.

Problem formulation

In this chapter we consider the following linearly constrained *mixed-integer nonlinear problem* (MINLP)

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f(x) \\ \text{s. to:} \quad & Ax \leq b, \quad A_{eq}x = b_{eq} \end{aligned} \tag{$\mathcal{P}$}$$

where f is a differentiable function with $L-$Lipschitz gradient (we will simply say that it is $L-$*smooth*). Here A, A_{eq}, b, b_{eq} are matrices and vectors of appropriate size, we consider that all in all there are $m = m_{\text{in}} + m_{\text{eq}}$ scalar constraints, where m_{eq} is the number of equality constraints and m_{in} the number of inequality constraints. We denote $x^\star$ an optimal point, and $\mathcal{X}^\star$ the optimal set, which we assume non-empty. We will also take the hypothesis that f is bounded below on $[0,1]^p$.

Applications

MINLPs occur across a multitude of applications. They arise frequently in power systems [149], transportation [89], logistics [110] and robotics [57]. For instance, discrete time hybrid systems can be approximated as mixed logical dynamic systems [25], which is a MINLP. In the field of machine learning, problems such as Boolean least squares, two-way partitioning problems or maximum cut [121] also fall under this general setting.

Tractability

Unfortunately, MINLPs are NP-hard in general as they include *mixed-integer linear programs* (MILP) as a special case. In fact, finding polynomial time algorithms for this class of problems is highlighted as one of the Millennium Prize Problems: $P \stackrel{?}{=} NP$. In this chapter, we develop approximate methods that converge in polynomial time.

Candidate point

A *candidate point*, $x \in \mathbb{R}^p$, aims at being close to the set of optimal solutions $\mathcal{X}^\star$. We do not require a candidate point to be feasible, as feasibility is often NP-hard itself. Given a particular application, some metrics can be defined to assess the quality of a given candidate as it is done in Section 5.5.

Literature review

We divide the various approaches to tackle $(\mathcal{P})$ into four categories: global methods, metaheuristics, convex approximations and structured heuristics. *Global methods* refer to methods that exactly solve $(\mathcal{P})$ and are often based on the branch-and-bound or branch-and-cut frameworks [23]. Efficient solvers have been built in the last decades, notably for MILPs and mixed-integer quadratic problems (MIQPs) with convex objective and constraints [23, 51]. Nevertheless, ultimately these methods have a complexity that grows exponentially with p, and are therefore limited as the problem grows in size. *Metaheuristics* include simulated annealing [88], tabu search [59], genetic algorithms [75], pchapter swarm optimization [85] and more. The reader can refer to [134] for a comprehensive review of these methods. *Convex approximations* methods consist in approximating $(\mathcal{P})$ with a convex problem. These include: binary relaxation when the objective and constraints are convex [127, 102], the Lagrangian relaxation technique [140, 121, 159] which is particularly appealing when the objective and constraints are quadratic because the corresponding dual problem can be explicitly formulated as a concave semidefinite problem [103, 60, 121]. Alternatively, in the quadratic setting, the problem can be directly relaxed to a semidefinite problem using a method called lifting [121]. Other convex procedures have been used, e.g. the convex-concave approach [100, 139, 121], and alternating direction method of multipliers (ADMM) [121, 27]. We call *structured heuristics*, methods that take advantage of the MINLP problem structure. These methods

can be seen as being in between convex approximations and metaheursitics. Hopfield Neural Networks (HNNs) fall in that category. HNNs were introduced in [76, 77], and were initially used in machine learning as a way to memorize the state of data [160]. To this day, variants of HNNs are still in use, notably in Deep Learning with the so called *Boltzmann Machine*, a stochastic variant of HNNs. In 1985, Hopfield and Tank showed that HNNs structure and dynamics could be used to find a candidate solution to the traveling salesman problem [78, 162, 55]. In that setting, contrary to machine learning applications, the weights of the HNNs are not obtained via training. They are directly defined by the parameters of the problem [147]. Since then, HNNs have been used for combinatorial problems such as clustering [82], vertex cover [128] and knapsack problems [119]. We refer the reader to [101, 142] for a comprehensive review of Hopfield method applications.

Contributions

This chapter advances our understanding of Lagrangian duality and Hopfield methods to recover candidate solutions to $(\mathcal{P})$. This advanced understanding provides a new (or newly understood) tool for approximately solving large-scale combinatorial problems.

Chapter organization

In Section 5.2 we introduce HNNs and give some background on their stability and equilibria. In Section 5.3 we introduce our numerical methods based on HNNs and prove their convergence properties (all the proofs are gathered in the final section). In Section 5.5 we provide some numerical experiments to illustrate how our methods can be used to approximately solve combinatorial optimization problems.

5.2 Hopfield Neural Nets

In this section we provide the necessary background on HNNs models, stability and equilibria in the continuous-time case. We then show that HNNs are closely related to continuous mirror descent which gives a natural geometric interpretation to these models but also brings to light some of their limitations.

Hopfield Neural Networks

Let us consider the following optimization problem with quadratic objective

$$\min_{x\in\mathcal{X}} \quad \frac{1}{2}x^\top Qx + q^\top x$$

The corresponding HNN is an autonomous nonlinear dynamic system given by the ordinary differential equation (ODE)

$$\begin{cases} \dot{x}_H(t) &= -Qx(t) - q \\ x(t) &= \phi(x_H(t)) \end{cases}$$

with initial condition $x_H(t=0) \in (0,1)^p$, where x and $x_H \in \mathbb{R}^p$ are called the *state* and *hidden state* of the system. The function $\phi : \mathbb{R}^p \to [0,1]^p$ is defined component-wise

$$\phi : u \mapsto (\phi_1(u_1), \cdots, \phi_p(u_p))$$

where $\phi_1, \cdots, \phi_p$ are *activation functions*, introduced in more detail in the next Section.

We extend the classic HNN to non-quadratic objectives: let us consider the problem $(\mathcal{P})$ without linear constraints

$$\min_{x \in \mathcal{X}} \quad f(x) \tag{$\mathcal{P}_{\text{unc}}$}$$

We define the corresponding HNN to be

$$\begin{cases} \dot{x}_H(t) &= -\nabla f(x(t)) \\ x(t) &= \phi(x_H(t)) \end{cases} \tag{5.2}$$

with initial condition $x_H(t=0) \in (0,1)^p$. The extension of classic HNNs consists in replacing the linear dynamic in x of the hidden state by a nonlinear function of x, $-\nabla f(x)$.

Activation functions

We consider the *activation functions* ϕ_i, $i \in I_p$ to be real-valued, β_i- Lipschitz continuous, symmetric with respect to (w.r.t.) $(\frac{1}{2}, \frac{1}{2})$, monotonically non-decreasing, surjective in $[0,1]$ and differentiable almost everywhere. Moreover we consider *smooth activation* functions to be differentiable with $\tilde{\beta}_i$ Lipschitz continuous derivative. We consider β_i to be a hyperparameter. In the literature, β_i is often called the *temperature*, a terminology we will also adopt.

As a heuristic, $\beta_i = 1$ corresponds to an activation function approximating the projection on $[0,1]$. If $\beta_i \to \infty$, ϕ_i asymptotically approaches the projection $\mathcal{P}_{\{0,1\}}$ on the set $\{0,1\}$ w.r.t. the $\mathcal{L}^1$ Lebesgue space. For continuous variables we typically choose $\beta_i = 1$ and for binary variable $\beta_i >> 1$, e.g. $\beta_i = 100$. Figure 5.1 shows graphical representations for some activation functions and for two temperature values $\beta \in \{1, 5\}$. We focus on three activations: *tanh, sin* and *Piecewise-Linear (pwl)*. The tanh activation is the only one that is a bijection from $\mathbb{R}$ to $[0,1]$ and that is also differentiable. The sin activation is only surjective and differentiable. The pwl is not bijective and not differentiable at $1/2 \pm \{1/(2\beta)\}$.

All the results that will be derived in this chapter are also valid if the activation ϕ_i is chosen to be the identity, which is particularly pertinent for continuous optimization variables $x_i \in \mathbb{R}$ that are not constrained to $[0,1]$.

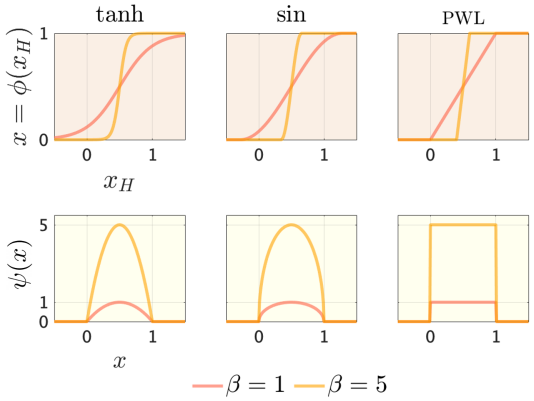

Figure 5.1: activation functions and corresponding gradient scaling $\psi(x) = \phi'(\phi^{-1}(x))$ for $\beta \in \{1, 5\}$

Stability

We now discuss the stability and equilibria of HNNs.

Theorem 5.2.1 (Decreasing objective function)**.** *The objective function f decreases along the unique state trajectory $x(t)$ of the HNN (5.2) - i.e. $t \to f(x(t))$ is a decreasing function.*

Hence, HNN is a descent trajectory for f and since we assumed the objective is bounded below on $[0, 1]^p$, we directly have convergence of $t \to f(x(t))$.

This is not surprising considering the fact that, if the activation is differentiable and bijective, the dynamic system (5.2) can be reformulated equivalently without the hidden state as

$$\dot{x} = -\psi(x) \odot \nabla f(x)$$

with $x(t = 0) \in (0, 1)^p$, where ψ is a *gradient scaling* vector defined as $\psi(x) := \phi'(\phi^{-1}(x))$ and for non bijective activations, $\psi(x) = \phi'(x_H)$. Graphical representations of gradient scaling w.r.t the activation function and temperature are given in Figure 5.1. This reformulation is also true for non-bijective activations as long as $x \in (0, 1)^p$. Nevertheless it is fundamental to understand that this equivalence does not hold for non-bijective activations as soon as there exists a component $i \in I_p$ such that $\phi_i(x_i) \in \{0, 1\}$. We can see from this formulation that the state dynamic is a scaled version of the gradient flow $\dot{x} = -\nabla f(x)$ as by construction $\psi \geq 0$.

HNN equilibria

From (5.2) the equilibrium points for x are included in

$$\mathcal{X}^{\dagger} := \prod_{i=1}^{p} \mathcal{X}_i^{\dagger}, \quad \text{where} \quad \mathcal{X}_i^{\dagger} = \{x \in [0,1]^p : \nabla f(x)_i = 0\} \cup \{0,1\} \tag{5.3}$$

The hidden state x_H might not have an equilibria and diverge if $\nabla f(x) \neq 0$ with $x \in \mathcal{X}^{\dagger}$. Also note the set $\mathcal{X}^{\dagger}$ is larger than the set of minimizers for problem $(\mathcal{P}_{\text{unc}})$ with binary relaxation,

$$\min_{x \in [0,1]^p} f(x) \tag{$\mathcal{P}_{\text{rel-unc}}$}$$

Using Karush-Kuhn-Tucker (KKT) conditions [26] for problem $(\mathcal{P}_{\text{rel-unc}})$, the set of local minimizers is included in

$$\mathcal{X}^{rel} := \prod_{i=1}^{p} \mathcal{X}_i^{rel} \quad \text{where} \quad \mathcal{X}_i^{rel} = \mathcal{X}_i^{\dagger} \cap \{\text{sgn}(x_i - 1/2)\nabla f(x)_i \leq 0\}$$

Hence $\mathcal{X}^{rel} \subset \mathcal{X}^{\dagger}$. This is a good trait for HNNs because a larger set of equilibria gives access to a larger set of candidate solutions. More precisely, the constraint $\text{sgn}(x_i - 1/2)\nabla f(x)_i \leq 0$ if $x_i \in \{0,1\}$ implies that the descent direction points outside the boundary of $\mathcal{X}$. The absence of this constraint is particularly appealing for binary constrained MINLPs as the descent direction might be pointing inside at the optimum, as illustrated in Fig. 5.2. An extensive analysis on the stability properties for x in $\mathcal{X}^{\dagger}$ is beyond the scope of this chapter. We refer the reader to [34] and [67] for a more in depth study on the equilibria stability for HNNs.

Stretching the phase portrait with temperature

The larger the temperature β_i, the larger the gradient scaling $\psi_i(x)$ for all $x \in [0,1]$ will be. More precisely, for the three activations presented in Figure 5.1 this relationship is proportional, i.e. $\psi_i(x) \propto \beta_i \psi_{\beta=1}(x)$. We can geometrically interpret it as follows: the choice of a high value for β_i stretches the phase portrait along the $i-$th component. The dynamic of the state is no longer perpendicular to the contour lines of f as is the case with gradient flow. Instead we can observe an angle that pushes the state to $\{0,1\}$. As an example let us consider the MINLP $(\mathcal{P}_{\text{unc}})$ with

$$f(x_1, x_2) = \frac{1}{4}(x_1 - 2/3)^4 + \frac{1}{2}(x_2 - 1/3)^2 + \frac{1}{3}x_1 x_2 \tag{5.4}$$

and $\mathcal{X} = \{0,1\} \times [0,1]$. Figure 5.2 illustrates this phase portrait stretching phenomenon. We can see that the red arrows that represent the instantaneous direction of x_H makes an angle with the gradient (black arrows) that favors

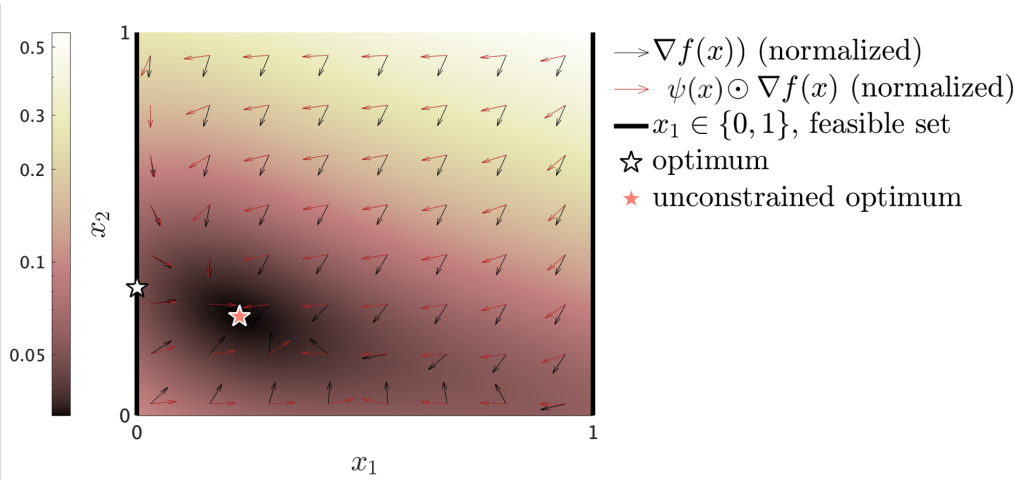

Figure 5.2: Normalized phase portrait of f defined in equation (5.4) with comparison between gradient flow and HNNs with tanh activation and temperatures $\beta_1 = 10$, $\beta_2 = 1$

HNNs and mirror descent

In [91], the authors show that continuous time mirror descent can be summarized by the following ODE

$$\begin{cases} \dot{x}_H &= -\nabla f(x) \\ x &= \nabla \Phi^\star(x_H) \end{cases} \tag{5.5}$$

with initialization $x_H(t = 0)$ and where Φ is a *mirror map* – by definition, (i) a differentiable and strictly convex function defined on a convex set $\mathcal{C}$ such that (ii) the range of $\nabla\Phi$ is $\mathbb{R}^p$ and (iii)$\nabla\Phi(x) \to \infty$ whenever x approaches the boundary of $\mathcal{C}$. If we choose $\nabla\Phi^\star = \phi$ in (5.5), then we get the same ODE (5.2) as HNNs. This also implies $\mathcal{C} = [0, 1]^p$ and $\Phi^\star = \int \phi + C$ (C a constant). Since ϕ is monotonically non-decreasing, $\Phi^\star$ is convex, and $\Phi = \Phi^{\star\star}$. If the activation is bijective then $\Phi = \int \phi^{-1} + C$ (C a constant). In that case, Φ is a mirror map. This equivalence between mirror descent and HNNs has also been proven in [68] using a different approach. If the activation function is not bijective, than we can still compute Φ in closed form, nevertheless it does not satisfy the conditions (ii, iii) for it to be a mirror map. This connection allows us to geometrically interpret HNNs: for instance mirror descent is often understood as an extension of Euclidean projection (introduced in more detail in the next section with the notion of Bregman divergence). We also refer the reader to [68] for an in-depth geometric view on HNNs. This connection allows us to use known results from continuous mirror descent: in [91, 41], the authors show that for f convex, the flow $x(t)$ from (5.5) converges to the minimum of f on $\mathcal{C}$. Hence, for bijective activations, we directly obtain the following result: HNN flow $x(t)$ converges to the optimal solution to $(\mathcal{P}_{\text{rel-unc}})$. Nevertheless, converging to $(\mathcal{P}_{\text{rel-unc}})$ is not a goal we have when using HNNs: with f convex,

we want a convex method that would favor convergence to the optimal set ($\mathcal{P}$). As mentioned earlier, the equivalence with mirror descent does not hold for all activations: in practice we discretize (5.2) in time, and, do not choose a numerical scheme that specifically enforces stability w.r.t. the ODE (5.2). All in all the choice of activation and step size leaves space for HNNs to converge to a candidate solution in $\mathcal{X}$.

Limitations of HNNs

As it is the case for most non-convex methods, convergence of HNNs depends on the initial conditions. This sensitivity can be partly explained by the fact that for $x \in (0,1)^p$, the HNNs dynamic is a scaled version of gradient flow. Therefore the flow $x(t)$ cannot cross the hypersurfaces $F_i = \{x \in \mathbb{R}^p : \nabla f(x)_i = 0\}$, $\forall i \in I_p$. Crossing F_i can be wanted if the optimal solution lies on the other side of it. We illustrate this with the same example as in Section (5.2). Figure 5.3 builds on Figure 5.2 by displaying the curve $F_1 = \{x : \nabla f_1(x) = 0\}$ and the flow $x(t)$, as well as a analytically tailored *descent* trajectory that crosses F_1 and does converge to the optimum. In the next section we will develop methods that can produce similar trajectories.
The dependence on the initial condition is also due to HNN being a descent method (Theorem 5.2.1): if $f(x(t=0)) < f(x^\star)$ then the trajectory will not be able to converge to $x^\star$. In other words, the HNN trajectory always stays within the set $\{x \in \mathbb{R}^p : f(x) \leq f(x(t=0))\}$. That is why we will propose in the next section to choose the initial point $x(t=0)$ by running a gradient ascent as a first step.

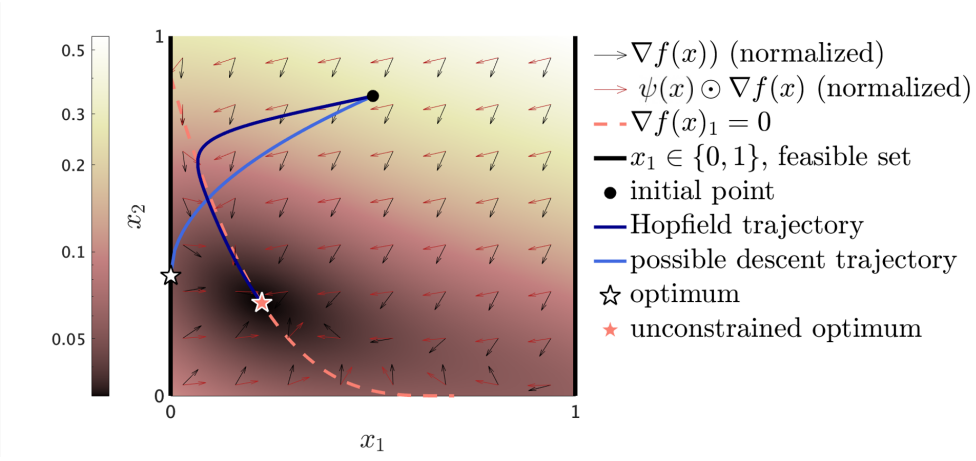

Figure 5.3: Normalized phase portrait of f defined in equation (5.4) with comparison between gradient flow and HNNs with pwl activation and temperatures $\beta_1 = 10$, $\beta_2 = 1$

5.3 Hopfield methods

In this section we develop numerical methods based on HNNs and its corresponding ODE (5.2). We recover all the properties of HNNs while tackling some possible issues and limitations highlighted in the previous section.

Explicit discretization scheme

The explicit discretization of HNNs (5.2) reads

$$\begin{cases} x_H^{k+1} &= x_H^k - \gamma^k \nabla f(x^k) \\ x^k &= \phi(x_H^k) \end{cases} \tag{5.6}$$

with initialization $x_H^0 \in (0,1)^p$ and discretization step size γ^k. We refer to equation (5.6) as the *Hopfield method*. We call *Hopfield methods*, the methods that generalize or build on (5.6).

In the following we will denote the positive diagonal matrix $\Psi^k := \mathrm{diag}(\phi'(x_H^k)) \succeq 0$.

Hopfield descent

We can directly extend the Hopfield method (5.6) by considering a more general direction d instead of $-\nabla f(x)$,

$$\begin{cases} x_H^{k+1} &= x_H^k + \gamma^k d^k \\ x^k &= \phi(x_H^k) \end{cases} \tag{5.7}$$

with initialization $x_H^0 \in (0,1)^p$. We call this method *Hopfield descent*, which can be used to tackle the issue of hypersurface crossing mentioned in Section (5.2). In order to guarantee the existence of a step-size γ^k such that $f(\phi(x^k + \gamma^k d^k)) \leq f(x^k)$ we define the *Hopfield descent condition* that constrains the direction d with respect to the gradient,

$$\nabla f(x^k)^\top \Psi^k d^k < 0 \tag{5.8}$$

In other words we require the descent direction d^k to make an acute angle (at most 90°) with the scaled descent $-\Psi^k \nabla f(x^k)$. In practice we will adopt a descent angle constraint parametrized by $\theta \in [0, \frac{\pi}{2})$ such that

$$n\big[\Psi^k \nabla f(x^k)\big]^\top d^k \leq -\cos(\theta)$$

with $\|d^k\| = 1$. Or equivalently,

$$\nabla f(x^k)^\top \Psi^k d^k \leq -\cos(\theta) \|\Psi^k \nabla f(x^k)\| \|d^k\| \tag{5.9}$$

The Hopfield descent offers the possibility to choose various descent direction d across iterations: we will present one way of making a judicious choice, given the combinatorial problems structure at hand, for a descent d^k in Section 5.3.

Descent violation

We also consider numerical methods that allow the condition $f(x^{k+1}) \leq f(x^k)$ to be violated. This can be done in order to favor $x_i^k \in \{0,1\}$ for $i \in I_b$. For instance, we set $\varepsilon > 0$ such that if $\psi(x^k)_i < \varepsilon$ (which implies that x_i^k is close to $\{0,1\}$), we set $x_i^{k+1} = \mathcal{P}_{\{0,1\}}(x_i^k)$ and $x_{H,i} = \infty$. By doing so, we stop the evolution of the state component x_i. Although not discussed in the chapter, stochastic directions can also be used in order to avoid local minima and reduce dependence on the initial condition. Such methods often lead to the descent condition being violated. As mentioned earlier, stochastic versions of HNNs are referred to as Boltzmann machines. Hopfield methods that use these techniques are studied in [68].

Initialization

In order to reduce the dependence on the initial condition where the state trajectory is trapped in the set $\{x \in \mathbb{R}^p : f(x) \leq f(x_0)\}$, we propose to initially run gradient ascent starting with $x_0 = 1/2$. That is, $x_0 \leftarrow x_0 + \gamma \nabla f(x_0)$ while $x_0 \in [\varepsilon, 1-\varepsilon]^p$, where ε is a hyperparameter. This creates a larger invariant set for Hopfield descent to evolve within. Alternatively we run the scaled gradient ascent $x_0 \leftarrow x_0 + \gamma b \odot \nabla f(x_0)$ where $b \in \mathbb{R}^p$ such that $b_i = 0$ if $i \in I_b$ and $b_i = 1$ otherwise. This keeps $x_i = 1/2$ for $i \in I_b$, and as a consequence does not bias convergence towards 0 or 1.

Connections with Projected gradient descent and mirror descent

Projected gradient method reads

$$\begin{cases} x_H^{k+1} & = x^k - \gamma^k \nabla f(x^k) \\ x^k & = \phi(x_H^k) \end{cases} \tag{5.10}$$

where ϕ is the piecewise linear (pwl) activation with $\beta = 1$. Contrary to Hopfield method, here the hidden state x_H is used as a temporary variable. That is, x_H does not have dynamics of its own. The proof of convergence for projected gradient descent relies on the non-expansiveness (i.e. 1-Lipschitz property) of convex projections [20]. Contrary to Hopfield methods, it is therefore not possible to choose $\beta_i >> 1$ to approximate the binary projection $\mathcal{P}_{\{0,1\}}$ and achieve provable convergence. Projected gradient descent (5.10) is equivalent to the proximal method

$$x^{k+1} = \underset{x \in [0,1]^p}{\operatorname{argmin}} \nabla f(x^k)^\top (x - x^k) + \frac{1}{2\gamma^k} \|x - x^k\|^2$$

Mirror descent consists in replacing the proximal term $1/2\|x - x^k\|^2$ with the *Bregman divergence* $\mathcal{D}_\Phi(x, x^k)$. As defined in [20], the Bregman divergence of a function Φ (assumed strictly convex and differentiable) is given by

$$\mathcal{D}_\Phi(u, v) = \Phi(u) - \big(\Phi(v) + \nabla \Phi(v)^\top (u - v)\big)$$

The Bregman divergence is non-negative and convex in the first argument (not always w.r.t the second argument) and in general lacks symmetry. Locally, Bregman divergence is a quadratic measure

$$\mathcal{D}_\Phi(u, u+\varepsilon) = 1/2\varepsilon^\top \nabla^2\Phi(u)\varepsilon + \mathcal{O}(\varepsilon), \quad \varepsilon \in \mathbb{R}^p$$

If we take $\Phi(u) = 1/2\|u\|^2$, then $\mathcal{D}_\Phi$ is the euclidean distance. In that sense, projected gradient is a particular case of mirror descent. The first order optimal condition for unconstrained mirror descent is

$$\nabla f(x^k) + \frac{1}{\gamma^k}(\nabla\Phi(x^{k+1}) - \nabla\Phi(x^k)) = 0 \iff \begin{cases} x_H^{k+1} = x_H^k - \gamma^k \nabla f(x^k) \\ x_H^k = \nabla\Phi(x^k) \end{cases}$$

Taking $\phi^{-1} = \nabla\Phi$ directly yields the Hopfield method (5.6). As in Section 5.2, the equivalence between Hopfield method and mirror descent requires a bijective activation. For the tanh activation, we have that $\Phi(x) \propto_x x\log(x) + (1-x)\log(1-x)$, the Bit entropy. This leads to a natural interpretation of the Hopfield method: x represents a probability of the solution being 1 or 0. As shown in [172], mirror descent will converge to the optimal solution of problem ($\mathcal{P}_{\text{rel-unc}}$) for Φ σ_Φ-strongly convex (i.e. $\nabla^2\Phi \succeq \sigma_\Phi I$ for twice differentiable Φ), if the step-size is chosen to be less than σ_Φ/L with a worst case rate of convergence $O(1/k)$. As presented in our convergence analysis, we will not specifically require the activation to be bijective or the step size to be such that $\gamma^k \leq \sigma_\Phi/L$: which leaves space to converge to points out of ($\mathcal{P}_{\text{rel-unc}}$).

Convergence of Hopfield methods

In this section we prove convergence for the Hopfield methods (5.6, 5.7). We start by showing that given a small enough step-size, the objective decreases at each iteration (this is the numerical equivalent of Theorem 5.2.1). We then prove a convergence rate when f is convex.

Theorem 5.3.1. *The Hopfield descent method (5.7) with a smooth activation function yields $f(x^{k+1}) \leq f(x^k)$, for the explicit step-size condition $\gamma^k \leq \Gamma_{f,\phi}(x^k, x_H^k, d^k)$, where the function $\Gamma_{f,\phi}$ is defined in equation (5.17) .*

From the proof of this theorem, choosing $\gamma^k = \Gamma_{f,\phi}(x^k, x_H^k, d^k)$ we have

$$\Delta f(x)^k \leq \frac{1}{2}\gamma^k \nabla f(x^k)^\top \Psi^k d^k \tag{5.11}$$

This leads to the following statement,

Corollary 5.3.1.1. *There exist $f^\dagger$ such that $f(x^k) \to f^\dagger$. Moreover, if x^k converges to the set $\mathcal{X}^\dagger$ defined in (5.3), then $f(x^k)$ also converges.*

As in Section 5.2, by direct analysis of the Hopfield method we have that the set of equilibria is $\mathcal{X}^\dagger$.
If we choose $d^k = -\nabla f(x^k)$, then we get the step size

$$\gamma^k = \Gamma_{f,\phi}(x^k, x_H^k) = \frac{\|\nabla f(x^k)\|_{\Psi^k}^2}{L\|\beta \odot \nabla f(x^k)\|^2 + \|\tilde{\beta}^{1/3} \odot \nabla f(x^k)\|_3^3}$$

If ϕ is the identity, then $\beta = 1$, $\tilde{\beta} = 0$, $\Psi^k = I$ then we get $\gamma^k = 1/L$, which is the classical steepest descent step size for smooth unconstrained optimization [20]. In that sense, Hopfield method is a generalization of the gradient descent algorithm. We also remark that this step size is not constant: It can go to ∞ as $\nabla f \to 0$ as the term $\|\nabla f\|_3^3$ dominates; or it can go to zero as $x \to \{0,1\}^p$. In that respect, Hopfield method is different from the constant step-size of mirror descent presented in Section 5.3. For bijective activations, although there is an equivalence with mirror descent, we argue that is not designed to guarantee convergence to the optimum of $(\mathcal{P}_{\text{rel-unc}})$ when the objective is convex. Importantly, this added flexibility (choice of descent direction) of Hopfield methods in discrete time offers the possibility of converging to sub-optimal points of $(\mathcal{P}_{\text{rel-unc}})$ that can be optimal points for the original MINLP problem $(\mathcal{P}_{\text{unc}})$.

Although we already have convergence, We would like to estimate the worst case scenario for convergence speed. It is not possible to do so without some additional assumptions on f. In Theorem 5.3.2 we derive a convergence rate under the condition that f is convex.

Theorem 5.3.2. *If the objective f is convex, then the Hopfield method* (5.7) *with (i) smooth activation function ϕ, (ii) verified Hopfield descent condition (5.8), and (iii) the component-wise condition*

$$\psi(x) \geq \beta \odot \min(|x|, |1-x|), \ \forall x \in [0,1]^p$$

yields a worst-case convergence speed that is sub-linear and $f(x^k) - f^\dagger = \mathcal{O}(\frac{1}{k^r})$ for all $r \in (0,1)$.

From this theorem, we show that the rate of convergence is similar to that of (projected) gradient descent as shown in [116] for the same hypothesis on f. In that regard, the Hopfield method is a relatively efficient algorithm.

As it is the case for classic gradient descent, the step-size (5.17) from Hopfield methods requires an estimate of the objective function's smoothness value L. In some applications we either do not have access to L, or we only have a conservative estimate of L which might slow down convergence. To tackle that issue, we propose a variant of the Armijo rule [20] that is tailored to our methods. Namely, let $\gamma > 0$ be a default step-size, at a given iteration k, we iterate $j \leftarrow j + 1$ until $\gamma^k = 2^{-j}\gamma$ is such that

$$\Delta f(x)^k \leq \sigma \gamma^k \nabla f(x^k)^\top \Psi^k d^k$$

where $\sigma \in (0, 1/2]$ is a fixed hyperparameter, usually chosen to be in $[10^{-5}, 10^{-1}]$ for classic nonlinear methods [20]. From equation (5.11), the Armijo rule is well-posed (i.e. j is finite) for $\sigma \leq 1/2$.

Choice of descent direction

In this section, we discuss how we choose the descent direction d^k for the Hopfield descent method in (5.7). At iteration k, let us define

$$\begin{cases} g^k &= -n[\Psi^k \nabla f(x^k)] \\ h^k &= -n[\nabla f(x^k)] \end{cases}$$

and $b^k \in \mathbb{R}^p$ a vector that points towards the closest point in the feasible set $\mathcal{X}$ (defined in (5.1)) for binary indices $i \in I_b$ given x^k. For instance, for $i \in I_b$, we can take $b_i^k = -1$ if $|x_i^k| < |x_i^k - 1|$ and $b_i^k = 1$ otherwise and for $i \in I_p$. Alternatively to avoid a high dependence on the initial state x_0, we can choose $b_i^k = \phi_i(x_i^k - \frac{1}{2})$, for $i \in I_b$. In both cases, we then proceed to normalize vector b, $b \leftarrow n[b]$.

Our objective is now to find a direction d^k that is as close as possible to b^k and that respects the Hopfield descent constraint (5.8). To do so we formulate a convex optimization problem: we minimize the Euclidean distance between the binary direction b^k and the scaled steepest descent h^k using a trade-off criterion. Dropping the iteration k superscript for clarity, the corresponding optimization problem reads

$$\begin{aligned} &\min_{d \in \mathbb{R}^p} \zeta \|d - b\|^2 + (1 - \zeta)\|d - h\|^2 \\ &\text{s. to:} \quad g^\top d \geq \cos(\theta), \quad \|d\| \leq 1 \end{aligned} \tag{5.12}$$

where $\zeta \in [0, 1]$ is the trade-off hyper-parameter. The optimization problem (5.12) is convex and can be solved analytically. Let us denote $w = \zeta b + (1 - \zeta)h$. Slater's condition holds, and thus strong duality holds, implying that the problem (5.12) is equivalent to its dual problem

$$\min_{y \geq 0} \Big(\max_{\|d\| \leq 1} d^\top (w + yg) \Big) - y \cos(\theta) \tag{5.13}$$

Using the fact that the convex conjugate of the Euclidean norm is the Euclidean norm itself gives us

$$\max_{\|d\| \leq 1} d^\top (w + yg) = \|w + yg\|$$

Then, using the first order condition for the convex problem (5.13), and solving a quadratic equation, we get the following formula for the dual variable y

$$y = \max \Big\{ 0, -g^\top w + \cot\theta \sqrt{\|w\|^2 - (g^\top w)^2} \Big\}$$

where cot denotes the *cotangent*, we then get the optimal direction

$$d = n[w + yg]$$

Hence, we have that $d = w$ if the Hopfield descent condition is satisfied for w as $y = 0$. This idea for finding a direction d^k is similar to the Frank-Wolfe algorithm [116, 52] and is only pertinent as long as there exists $i \in I_b$ such that x_i^k is not in $\{0, 1\}$. When $x_i^k \in \{0, 1\}$ we can simply take the direction to be the opposite of the gradient.

5.4 Dual Hopfield methods

So far we have only dealt with combinatorial problems without linear constraints. In this section, we want will focus on methods for constrained MINLP ($\mathcal{P}$). In the literature for Hopfield methods for constrained optimization [78, 162, 55], the linear equality constraint are relaxed and included as a penalty term. This is referred to as *pure penalty approach*. It consists in solving

$$\min_{x \in \mathcal{X}} \quad f(x) + \frac{\rho}{2} \|A_{eq}x - b_{eq}\|^2 \tag{5.14}$$

where $\rho > 0$ is a hyperparameter tuned by the practitioner. The main drawback of this method is the inexistence of a finite ρ such that problems ($\mathcal{P}$) and (5.14) yield equivalent solutions, even in the strictly convex case with binary relaxation, as shown in [20] (Section 3.1.1), the parameter ρ has to be taken to $+\infty$. To achieve a given precision on constraint violation, ρ has to be taken large, which creates numerical instability and generally slows down convergence [116, 20]. In order to tune ρ, trial and error approaches such as the sequential unconstrained maximization technique [109] have been proposed. Different penalty approaches have been used for Hopfield methods: in [167] the authors propose one hyperparameter per scalar linear equality, in [3] the penalty is scaled by a matrix, via a technique called the *subspace approach*, which only works for linear equality constraints. All in all these methods never entirely solve the aforementioned issue. Our approach to modeling linear constraints consists in using augmented Lagrangian methods.

Lagrangian duality

For clarity and without loss of generality we can combine the linear equality and inequality constraints in a single linear inequality constraint that we still write $Ax \leq b$. The augmented Lagrangian for problem ($\mathcal{P}$) then reads,

$$\mathcal{L}_\rho(x, \mu) = f(x) + \mu^\top (Ax - b) + \frac{\rho}{2} \|(Ax - b)_+\|^2$$

where $(.)_+ = \max(0, .)$ is the projection on the positive orthant. We have by assumption on f that $x \to \mathcal{L}(x, \mu)$ is smooth, because the penalty $\frac{\rho}{2}\|(Ax - b)_+\|^2$ is differentiable w.r.t. x with gradient $\rho A^\top (Ax - b)_+$, which is Lipschitz continuous with Lipschitz constant bounded by $\rho \|A\|_2^2$. The dual function q is concave (by construction) and is given by the optimization problem

$$q(\mu) = \min_{x \in \mathcal{X}} \mathcal{L}_\rho(x, \mu) \tag{$\mathcal{D}_\mu$}$$

The dual problem is then the following optimization problem

$$\max_{\mu \geq 0} q(\mu)$$

Finding the optimal set, or a point μ in this set is challenging, notably because it requires solving the MINLP problem ($\mathcal{D}_\mu$) for many dual variables μ: for instance the dual (subgradient)

ascent algorithm reads

$$\mu^{k+1} = \left(\mu^k + \gamma^k \nabla_x \mathcal{L}_\rho(x(\mu), \mu)\right)_+ \tag{5.15}$$

where γ^k is a step-size and $x(\mu)$ an optimal solution to $(\mathcal{D}_\mu)$. In this algorithm we leveraged the fact that the augmented Lagrangian is differentiable w.r.t. x and that from Danskin's theorem [20] (proposition B.22) we have $\nabla_x \mathcal{L}_\rho(x(\mu), \mu) \in \partial q(\mu)$.
The dual problem $(\mathcal{D}_\mu)$ is of the same type as problem $(\mathcal{P}_{\text{unc}})$: it is an unconstrained MINLP with smooth objective. Instead of computing $x(\mu)$ exactly we can run a Hopfield method, as presented in section 5.3, to obtain an approximate solution $x^\dagger(\mu)$. We then replace $x(\mu)$ by $x^\dagger(\mu)$ in the dual ascent update (5.15). In the next section, we also propose another method for finding the dual variable μ when the objective f is convex or when its bi-conjugate can be obtained.

Lagrangian duality and convexified problem

The convex relaxation for $(\mathcal{P})$ is

$$\begin{aligned} \min_{x \in [0,1]^p} \quad & f^{\star\star}(x) + \frac{\rho}{2}\|(Ax - b)_+\|^2 \\ \text{s. to:} \quad & Ax \leq b \end{aligned} \tag{$\mathcal{P}_{\text{conv}}$}$$

where $f^{\star\star}$ is the bi-conjugate of f. It is know that the bi-conjugate is also the convex envelope of f [20]. A quadratic penalty term is added here, nevertheless as the problem is convex, all problems are equivalent for any $\rho \geq 0$. Remark that if the objective is already convex then $f = f^{\star\star}$. In cases for which $f^{\star\star}$ can be obtained analytically, or approximated tractably, we propose, alternatively to the previous section, to compute the optimal dual variable by solving the convex problem $(\mathcal{P}_{\text{conv}})$. We support this method via the following theorem.

Theorem 5.4.1. *Problem $(\mathcal{P})$ and its convexified counterpart $(\mathcal{P}_{conv})$ with $\rho = 0$ share the same dual function. And have therefore the same set of optimal dual variables.*

Based on this theorem we propose to compute an optimal dual variable μ by solving $(\mathcal{P}_{\text{conv}})$ either with a given solver or via the method of multipliers [20]. We can then proceed by running an Hopfield method to solve $(\mathcal{D}_\mu)$ with an optimal μ as a last step.

5.5 Numerical experiments

LogSumExp objective

We start by illustrating our method on the following optimization problem

$$\min_{x \in \mathcal{X}} \log(1_m^\top \exp(Ax + b))$$

where $A \in \mathbb{R}^{m \times p}$ was drawn randomly with distribution $\text{Unif}[-5/2, +5/2]^{m \times p}$ and b was drawn from $\text{Unif}[-5, +5]^m$. We also chose $ub = +10$ and $lb = -10$, each component x_i was constrained be binary randomly using a Bernoulli distribution with probability $1/2$. Note that the objective function is convex and smooth (the gradient of LogSumExp being the softmax, which is 1-Lipschitz [54]). We compare three methods: (a) Hopfield descent using the technique described in section 5.3 with $\zeta = 0.9$ and $\theta = \pi/2 - 0.1$, (b) Hopfield method (5.6) with $\beta_i = 10^2$ for $i \in I_b$ and $\beta_i = 1$ otherwise, (c) projected gradient descent (5.10). We run $K = 10^4$ iterations for each method. In order to compare the final result of these three methods, we project the final candidate solution x^K onto $\mathcal{X}$ and evaluate the corresponding objective value (dashed lines in the figure) .We illustrate in figure 5.4 the convergence curve of $f(x^k)$ with respect to the iteration k. We can see, as expected, that the convergence is slower for method (a) as it does not use the steepest descent $-\nabla f(x)$ and for $K = 10^4$, $f(x^K_{(c)}) < f(x^K_{(b)}) < f(x^K_{(a)})$. Nevertheless, (a) converges to a point in $\mathcal{X}$ whereas the two other methods don't, which results in a better candidate solution than the two other methods, it turns out that $f(\mathcal{P}_{\mathcal{X}}(x^K_{(c)})) > f(\mathcal{P}_{\mathcal{X}}(x^K_{(b)})) > f(\mathcal{P}_{\mathcal{X}}(x^K_{(a)}))$. In the end, the Hopfield method (b) also outperforms projected gradient descent. We want to stress out, that there exist instances for which projected gradient descent outperforms Hopfield, nevertheless the main point of this example is to illustrate that Hopfield method is better at finding candidate solution close to the feasible set $\mathcal{X}$. In the next section, we do compare Hopfield methods to other methods more thoroughly.

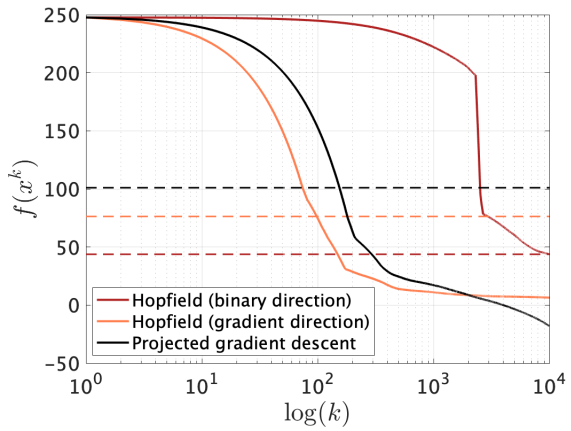

Figure 5.4: $f(x^k) = \text{LogSumExp}(Ax^k + b)$ across iterations, dashed lines are the value of the objective after projection on $\mathcal{X}$ for $k = 10^4$ and for each method: (a) Hopfield (binary direction), (b) Hopfield (gradient direction), (c) projected gradient descent

Application to economic load dispatch

In this example, we illustrate Dual Hopfield. Let us consider an electricity systems with m generators. At an initial time $t = 0$, a generator $i \in [n]$ status is either 'on' or 'off'. We gather in a vector $s_0 \in \mathbb{R}^m$ these initial status, with $s_{0,i} = 1$ if 'on' and $s_{0,i} = 0$ otherwise. We also denote $y_{0,i} \geq 0$ the initial power output of the generators at initial time. The agent responsible for the economic dispatch (called the ISO) has to make a decision for $t = 1$ on whether a given generator should be turned 'on' or 'off' as well as its power output in order to satisfy a power demand $D \geq 0$. This decision is made based on the following cost function

$$f(x = [z, y]) = p^\top y + \frac{1}{2} y^\top \Sigma y + \frac{C_1}{2} \|y - y_0\|^2 + \frac{C_2}{2} \|z - z_0\|^2$$

where $p \in \mathbb{R}^m_+$ is the vector of production marginal prices, $\Sigma \succeq 0$ is here to represent the uncertainty on prices (this is often referred to as Markowitz portfolio optimization). The two quadratic terms represent the cost from having to change the power output and status of the generator. The set mixed-binary set $\mathcal{X}$ is here defined as $\mathcal{X} = \{x = [z, y] \mid z \in \{0, 1\}^m$ and $y \in [0, 1]^m\}$. Moreover the ISO needs to satisfy the two following constraints, $1_m^\top y = D$ (supply equals demand) and the fact that a generator that is off cannot produce electricity which can be formulated as $z^\top y = D$. This equality constraint is not linear, nevertheless we will show that the Hopfield dual ascent method presented in section 5.4 still allows to find good candidate solutions. We define the augmented Lagrangian as,

$$\mathcal{L}_\rho(z, y, \lambda_1, \lambda_2) = f(z, y) + \lambda_1(1_m^\top y - D) + \lambda_2(z^\top y - D) + \frac{\rho}{2}(1_m^\top y - D)^2 + \frac{\rho}{2}(z^\top y - D)^2$$

We initialize the dual variables $\lambda_1 = \lambda_2 = 0$ and then proceed by running an Hopfield method (5.6) with sin activation and activation parameter $\beta = [10.1_m; 1_m]$. We then do a dual update

$$\lambda_1 = \lambda_1 + \gamma_D(1_m^\top y - D), \quad \lambda_2 = \lambda_2 + \gamma_D(z^\top y - D)$$

where λ_D is the dual step size. We then repeat these steps until convergence. For comparison we solve the convex quadratic problem

$$\begin{aligned} \min_y \; & p^\top y + \frac{1}{2} y^\top \Sigma y + \frac{C_1}{2} \|y - y_0\|^2 \\ & 1_m^\top y = D \\ & 0 \leq y \leq 1 \end{aligned}$$

and then simply round y with precision ϵ and define $z = 1(y \geq \epsilon)$ where ϵ is a precision parameter (we choose $\epsilon = 10^{-3}$). By construction the candidate solution obtained from this method $x = [z, y] \in \mathcal{X}$. We draw the parameters $\{p, \Sigma, C_1, C_2.D\}$ randomly and solve 100 instance of it with the Hopfield and convex method for $m = 20$ generators. We evaluate the 'supply equals demand' constraint violation with the criterion $\log_{10}(|y^\top z - D|)$ and the binary constrain violation with the criterion $z \odot (1 - z)/m$. From the simulations we have

that the binary constraint violation obtained with Hopfield is low: on average 7.4×10^{-4}, the distribution of which is highly concentrated around this value as it can be seen in figure 5.5. The dual method allows to satisfy the constraint with a mean precision of $10^{-2.5}$ which is close to the working precision we use for the convex solution $\epsilon = 10^{-3}$. In order to understand what these precision values represent, relatively the average the demand is $\mathbb{E}[D] = 5.07$ with a standard deviation $\sigma(D) = 1.63$. As it can be seen in figure 5.5, the Hopfield method outperforms the convex method in terms of cost, with an average of 2.48 for Hopfield versus 7.1 for the convex method.

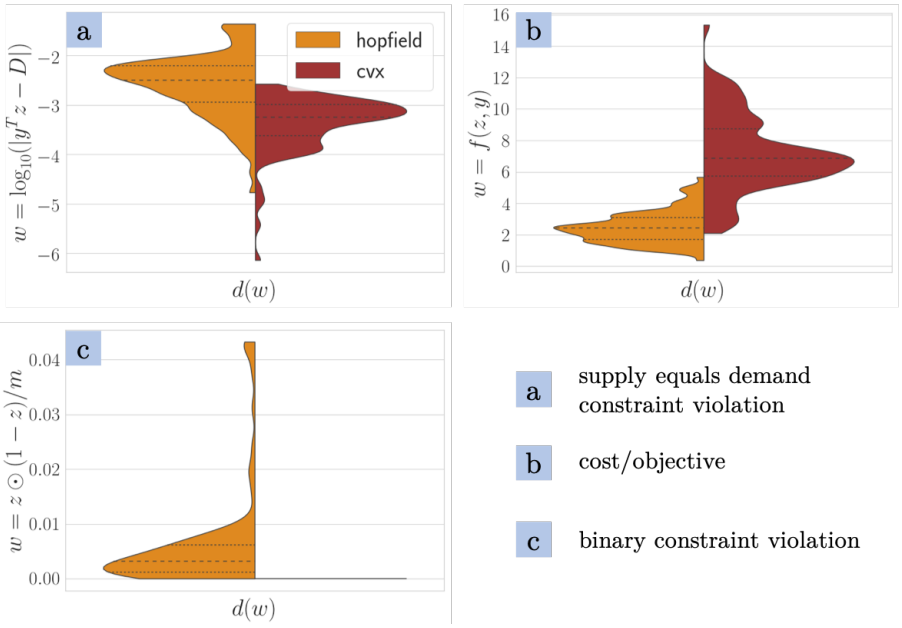

Figure 5.5: Violin plot – i.e. empirical probability density $d(w)$ on the x-axis w.r.t. w on the y-axis, where w is a criterion value – for (a) supply equals demand constraint violation, (b) the final cost, and (c) the binary constraint violation. The dashed lines represent the average and the quartile of the distribution.

5.6 Limitations

In this section, we remind the reader of the limitations associated with the methods that have been developed throughout this chapter.

First, we want to stress out the fact that Hopfield methods are heuristics, and that for this reason there is no guarantee of convergence to a solution to the original problem, let alone a feasible solution. That is why we called *candidate solution* the result of running Hopfield methods. The activation maps we propose do favor convergence to binary solutions, but

activations other than the projection on the binary set itself cannot guarantee convergence to such a set.

At the end of section 5.2 we already pointed out some limitations of HNNs, one of which was the sensitivity to the initial state that we attributed in part to the fact that HNNs are scaled version of gradient flow and cannot cross hypersurfaces where the components of the gradient is zero. With the descent method proposed later, we tackled part of this issue by allowing to cross such hypersurfaces. Nevertheless the problem of sensitivity to initial condition still holds.

In practice, Hopfield methods perform well. Nevertheless many parameters still need to be tuned given a problem at hand: for instance the allowable angle in the descent method, the choice of activation, or the choice of the dual update step size. Remark that even in the convex case, the choice of dual step size is challenging. Hence, for Hopfield methods, much like when learning with Neural Networks, the choice of step-size requires careful tuning.

All in all, the methods developed in this chapters should be seen as potential tools that can be used by a practitioner given on a combinatorial problem with constraints. The use of such methods require refined tuning of parameters such as the choice of activation functions and step sizes.

5.7 Proofs

Proof of Theorem 5.4.1

Problem $(\mathcal{P})$ can be equivalently reformulated as the following optimization problem

$$\begin{aligned} \min_{x\in\mathcal{X}, y\in\mathbb{R}^p} \quad & f(x) \\ \text{s. to: } & Ay \leq b, \quad x = y \end{aligned}$$

The corresponding dual problem is

$$\begin{aligned} \max_{\mu\geq 0,\lambda} \Big(& \min_{x\in\mathbb{R}^p}(\tilde{f}(x) + \lambda^\top x) + \cdots \\ & \min_{y\in\mathbb{R}^p}(\mu^\top \tilde{F}(y) - \lambda^\top y)\Big) \end{aligned}$$

Let us assume that at least one constraint is active at optimum, then we can assume $\mu \neq 0$ without loss of generality. Because X is a compact set, the conjugate functions of $\tilde{f}$ and $\mu^\top \tilde{F}$ are defined on $\mathbb{R}^p$. We have

$$\min_{x\in\mathbb{R}^p}(\tilde{f}(x) + \lambda^\top x) = -\tilde{f}^\star(-\lambda)$$

$$\min_{y\in\mathbb{R}^p}(\mu^\top \tilde{F}(y) - \lambda^\top y) = -(\mu^\top \tilde{F})^\star(\lambda)$$

Hence the dual problem reads

$$\max_{\mu\geq 0,\lambda} -\tilde{f}^{\star}(-\lambda) - (\mu^{\top}\tilde{F})^{\star}(\lambda)$$

which is equivalent to

$$\begin{aligned}\max_{\mu\geq 0,\lambda_1,\lambda_2} \quad & -\tilde{f}^{\star}(\lambda_1) - (\mu^{\top}\tilde{F})^{\star}(\lambda_2)\\ \text{s.to:} \quad & \lambda_1+\lambda_2 = 0\end{aligned}$$

The dual of this last problem is

$$\min_{x\in\mathbb{R}^p}\max_{\mu\geq 0}\Big(\max_{\lambda_1}\big(\lambda_1^{\top}x - \tilde{f}^{\star}(\lambda_1)\big) + \max_{\lambda_2}\big(\lambda_2^{\top}x - (\mu^{\top}\tilde{F})^{\star}(\lambda_2)\big)\Big)$$

which can be rewritten

$$\min_{x\in\mathbb{R}^p}\Big(\tilde{f}^{\star\star}(x) + \max_{\mu\geq 0}(\mu^{\top}\tilde{F})^{\star\star}\Big)$$

Using the properties of the conjugate

$$(\mu^{\top}\tilde{F})^{\star\star} = \mu^{\top}\tilde{F}^{\star\star}$$

We also have $f^{\star\star} = \tilde{f}^{\star\star}|_{Co(X)}$ and $F^{\star\star} = \tilde{F}^{\star\star}|_{Co(X)}$ where Co denotes the convex hull. We have that $Co(X) = [0,1]^p$. Then using the fact that $f^{\star\star} = \hat{f}$, all things considered, we get that the dual of the dual is the convexified problem ($\mathcal{P}_{\text{conv}}$). This result also holds if all the constraints are inactive as

$$\hat{F}(x) \leq F(x),\ \forall x \in \hat{X}$$

Proof of Theorem 5.2.1

equation (5.2) can be equivalently reformulated as

$$\begin{cases}\dot{x}_H &= -\nabla f(x)\\ \dot{x} &= -\phi'(x_H)\odot\nabla f(x)\end{cases}$$

with $x_H(t=0)\in(0,1)^p$ and $x(t=0) = \phi(x_H(t=0))$. Using this equivalent formulation, if ϕ IS smooth, Picard-Lindelöf theorem ([136], Ch.3) applies, and therefore this dynamical system is well-posed. Proving unicity of solution for non differentiable can be done using extension of Picard-Lindelöf [136]. Moreover we have the Lie derivative

$$\frac{d}{dt}f(x) = -\big(x_H\odot\phi'(x_H)\big)^{\top}x_H = -x_H^{\top}\text{diag}(\phi'(x_H))x_H$$

as $\text{diag}(\phi'(x_H))\geq 0$, we have $\frac{d}{dt}f(x)\leq 0$ which ends the proof.

Descent lemma and inequalities

Lemma 5.7.1. *(Descent lemma) Given h an L-smooth real-valued function,*

$$\forall u, v \quad h(u) - h(v) \leq (u-v)^\top \nabla h(v) + \frac{L}{2}\|u-v\|^2$$

The proof of this lemma can be found in [20] (proposition A.24).

Lemma 5.7.2. *(Framing lemma) Let $h : \mathbb{R} \to \mathbb{R}$ a real L-smooth function, then*

$$h'(v)(u-v) - \frac{L}{2}(u-v)^2 \leq h(u) - h(v) \leq h'(v)(u-v) + \frac{L}{2}(u-v)^2$$

Proof. Let $m(t) := h(tu + (1-t)v)$, then

$$\begin{aligned} h(u) - h(v) &= m(1) - m(0) \\ &= \int_0^1 m'(t)dt \\ &= \int_0^1 (u-v)h'(tu+(1-t)v)dt \\ &= h'(v)(u-v) + \int_0^1 (u-v)[h'(tu+(1-t)v) - h'(v)]dt \end{aligned}$$

Let us denote $q(t) := tu+(1-t)v$ from L-smoothness of h, we have $|h'(q(t))-h'(v)| \leq Lt|u-v|$ which implies

$$-Lt(u-v)^2 \leq (u-v)[h'(q(t)) - h'(v)] \leq Lt(u-v)^2$$

and therefore

$$-\frac{L}{2}(u-v)^2 \leq \int_0^1 (u-v)[h'(q(t)) - h'(v)]dt \leq \frac{L}{2}(u-v)^2$$

this concludes the proof. □

Proof of Theorem 5.3.1

Recall the following notation: given some sequence $\{u^k\}_k$, we denote $\Delta h(u)^k = h(u^{k+1}) - h(u^k)$. Using the descent Lemma 5.7.1, we have

$$\Delta f(x)^k \leq \underbrace{\Delta x^{k\top} \nabla f(x^k)}_{\text{Part I}} + \underbrace{\frac{L}{2}||\Delta x^k||^2}_{\text{Part II}}$$

We start by producing upper-bounds for Part I and II of this inequality.

Part I: using equation (5.7),

$$\Delta x^{k\top}\nabla f(x^k) = \Sigma_{i\in I_p}\Delta\phi_i(x_{H,i})^k\nabla f(x^k)_i \tag{5.16}$$

Now, we want to leverage the monotonicity and $\tilde{\beta}$ gradient Lipschitz continuity of the activation function to upper-bound $\Delta\phi_i(x_{H,i})^k\nabla f(x^k)_i$. For conciseness, let us drop the index i for now. Using lemma 5.7.2 we get

$$\phi'(x_H^k)\Delta x_H^k - \frac{\tilde{\beta}}{2}(\Delta x_H^k)^2 \leq \Delta\phi(x_H)^k \leq \phi'(x_H^k)\Delta x_H^k + \frac{\tilde{\beta}}{2}(\Delta x_H^k)^2$$

Two cases arise

- **case 1:** $\nabla f(x^k) \geq 0$, using equation (5.7) and the previous inequality

$$\Delta\phi(x_H)^k\nabla f(x^k) \leq \gamma^k d^k\psi^k\nabla f(x^k) + \frac{\tilde{\beta}}{2}(\gamma^k d^k)^2\nabla f(x^k)$$

- **case 2:** $\nabla f(x^k) < 0$, again, using equation (5.7) and the previous inequality

$$\Delta\phi(x_H)^k\nabla f(x^k) \leq \gamma^k\psi^k d^k\nabla f(x^k) - \frac{\tilde{\beta}}{2}(\gamma^k d^k)^2\nabla f(x^k)$$

In conclusion, in both cases, using the index i once again, we have

$$\Delta\phi(x_{H,i})^k\nabla f(x^k)_i \leq \gamma^k\psi_i^k d_i^k\nabla f(x^k)_i + \frac{\tilde{\beta}_i}{2}(\gamma^k d_i^k)^2|\nabla f(x^k)_i|$$

Going back to the full sum, using equation (5.16) leads to the inequality

$$\Delta x^{k\top}\nabla f(x^k) \leq \gamma^k\nabla f(x^k)^\top\Psi^k d^k + \frac{1}{2}(\gamma^k)^2 d^{k\top}\mathrm{diag}\big(\tilde{\beta}\odot|\nabla f(x^k)|\big)d^k$$

Part II of the inequality: using β-smoothness of the activation

$$||\Delta x^k||^2 = ||\Delta\phi(x_H)^k||^2 \leq \|\beta\odot\Delta x_H^k\|^2 = (\gamma^k)^2||\beta\odot d^k||^2 = (\gamma^k)^2 d^{k\top}\mathrm{diag}(\beta^2)d^k$$

where $.^2$ is seen as an component-wise operation on β.
Synthesis of Part I and II: gathering inequalities from Part I and II gives

$$\nabla f(x)^k \leq c_1^k\gamma^k + c_2^k(\gamma^k)^2$$

where $c_1^k = \nabla f(x^k)^\top\Psi^k d^k$ and $c_2^k = \frac{1}{2}d^{k\top}\mathrm{diag}\Big(\tilde{\beta}\odot|\nabla f(x^k)| + L\beta^2\Big)d^k$
we directly have $c_2^k \geq 0$ and from the Hopfield descent condition (5.8), $c_1^k < 0$, this is the motivation for this condition in the first place. This guarantees the existence of α^k such that $f(x^{k+1}) \leq f(x^k)$ at every iteration. The optimal step size is

$$\gamma^k = -\frac{c_1^k}{2c_2^k} = \frac{-\nabla f(x^k)^\top\Psi^k d^k}{d^{k\top}\mathrm{diag}\Big(\tilde{\beta}\odot|\nabla f(x^k)| + L\beta^2\Big)d^k} := \Gamma_{f,\phi}(x^k, x_H^k, d^k) \tag{5.17}$$

Proof of Corollary 5.3.1.1

Every bounded and monotonically nonincreasing sequence converge, therefore from Theorem 5.3.1 we have the existence of $f^\dagger \in \mathbb{R}$ such that $f(x^k) \to f^\dagger$, and therefore $\Delta f(x)^k \to 0$. Let us take, $\gamma^k = \Gamma_{f,\phi}(x^k, x_H^k, d^k)$. By smoothness of f, there exist $c > 0$ such that $\forall x \in [0,1]^p$,

$$d^{k\top}\mathrm{diag}\Big(\tilde{\beta} \odot |\nabla f(x^k)| + L\beta^2\Big)d^k \leq c\|d^k\|^2$$

Indeed, let $\tilde{c} \in \mathbb{R}$ such that $\tilde{c}1_p \geq |\nabla f(x)|$, $\forall x \in [0,1]^p$, then define $c = \tilde{\beta}\tilde{c} + L\|\beta\|^2$.
From that we have,

$$\gamma^k \geq \frac{-\nabla f(x^k)^\top \Psi^k d^k}{c\|d^k\|^2}$$

Injecting that inequality in (5.11) and using Hopfield descent condition (5.9) we get

$$\begin{aligned}\Delta f(x)^k &\leq -\frac{1}{2c\|d^k\|^2}\big(\nabla f(x^k)^\top \Psi^k d^k\big)^2 \\ &\leq -\frac{\cos^2(\theta)}{2c}\|\Psi^k \nabla f(x^k)\|^2\end{aligned} \tag{5.18}$$

Using this last inequality, we have that $\Delta f(x)^k$ converges to 0 when $\|\Psi^k \nabla f(x^k)\|$ converges to 0 (which is equivalent to, x^k converges to $\mathcal{X}^\dagger$). This proves the second part of the corollary.

Proof of Theorem 5.3.2

The convexity of the objective f gives the following inequality for all $x \in \mathcal{X}$

$$\begin{aligned}f(x^k) - f(x) &\leq \nabla f(x^k)^\top (x^k - x) \\ &\leq \|\nabla f(x^k) \odot (x^k - x)\|_1\end{aligned} \tag{5.19}$$

Let us define $\tilde{\mathcal{X}}^\dagger = \mathcal{X}^\dagger \cup \{x|\ f(x) = f^\dagger\}$, and let $\tilde{x}^k = \mathrm{argmin}_{x \in \tilde{\mathcal{X}}^\dagger}\|x - x^k\|$.
This leads to,

$$\min_{x \in \tilde{\mathcal{X}}^\dagger} f(x^k) - f(x) = f(x^k) - f^\dagger$$

and therefore using this with inequality (5.19)

$$\begin{aligned}f(x^k) - f^\dagger &\leq \min_{x \in \tilde{\mathcal{X}}^\dagger} \|\nabla f(x^k) \odot (x^k - x)\|_1 \\ &\leq \|\nabla f(x^k) \odot (x^k - \tilde{x}^k)\|_1\end{aligned} \tag{5.20}$$

As x^k converges to the set $\tilde{\mathcal{X}}^\dagger$ there exists an integer K such that for all $k \geq K$ and for all $i \in I_p$,

$$\min(|x_i^k|, |1 - x_i^k|) \geq |x_i^k - \tilde{x}_i^k|$$

From now on, let us assume $k > K$. Denoting $\delta f^k = f(x^k) - f^\dagger$,

$$b^k = \min(|x^k|, |1 - x^k|) \in \mathbb{R}^p$$

and $B^k = \text{diag}(b^k)$, we can rewrite inequality (5.20) as $\delta f^k \leq \|B^k \nabla f(x^k)\|_1$, and using the equivalence between $||.||_1$ and $||.||$

$$\delta f^k \leq \sqrt{p} \|B^k \nabla f(x^k)\| \tag{5.21}$$

We will be using this inequality later in the proof. From equation(5.18) we directly get,

$$\delta f^{k+1} \leq \delta f^k - \frac{\cos^2(\theta)}{2c} \|\Psi^k \nabla f(x^k)\|^2$$

Using the component-wise condition give in the statement

$$\psi^k \geq \beta \odot \min(|x^k|, |1 - x^k|) = \beta \odot b^k \geq \underline{\beta} b^k$$

with $\underline{\beta} = \min_{i \in I_p} \beta_i$, from this we get

$$\delta f^{k+1} \leq \delta f^k - \frac{\underline{\beta}^2 \cos^2(\theta)}{2c} \|B^k \nabla f(x^k)\|^2$$

Injecting inequality equation(5.21) we get,

$$\delta f^{k+1} \leq \delta f^k - \frac{\underline{\beta}^2 \cos^2(\theta)}{2cp} (\delta f^k)^2 \tag{5.22}$$

We know show that this inequality implies convergence of the sequence $\{\delta f^k\}$ at a rate $o(\frac{1}{k^r})$, with $0 < r < 1$. Let us define $\zeta = \frac{\underline{\beta}^2 \cos^2(\theta)}{2cp}$ and let us take the inverse of the inequality (5.22),

$$\begin{aligned}
\frac{1}{\delta f^{k+1}} &\geq \frac{1}{\delta f^k} \frac{1}{1 - \zeta \delta f^k} \\
&\geq \frac{1}{\delta f^k} \left(1 + \zeta \delta f^k\right) \\
&= \frac{1}{\delta f^k} + \zeta
\end{aligned}$$

Leveraging the telescopic sum over k (starting from index K) this inequality leads to

$$\frac{1}{\delta f^k} \geq \frac{1}{\delta f^k} + \zeta(k - K)$$

inverting this inequality and multiplying both sides by k^r gives

$$k^r \delta f^k \leq \frac{k^r}{\frac{1}{\delta f^k} + \zeta(k - K)}$$

The right hand side of this inequality is equivalent to $\frac{1}{\zeta} k^{r-1}$ when $k \to +\infty$, which converges to 0 for $r < 1$. As a consequence, $k^r \delta f^k \to 0$. More precisely, we have that $\delta f^k = O(\frac{1}{\zeta k})$, hence, to reach a precision ε such that $\delta f^k \leq \varepsilon$ we need a number of iterations $O(\frac{1}{\varepsilon})$.

Part III

Chapter 6

Implicit Optimization: Models and Methods

Let us start by defining what we mean when we use the word *implicit* in the two next chapters. As an example, we previously introduced the Hopfield Neural Network for a map f the following way

$$\begin{cases} \dot{x}_H(t) = -\nabla f(x(t)) \\ x(t) = \phi(x_H(t)) \end{cases}$$

We discretized this ODE in time with an explicit scheme and introduced the first versions of Hopfield methods. The original ODE is implicit: there is no analytical solution that can be derived to get the maps $\{x(\cdot), x_H(\cdot)\}$. Generally speaking, this is the case for most solutions to PDEs and ODEs. Most of the time, we form approximate solutions using (explicit or implicit) stable numerical schemes. Depending on the implicit problem at hand there are different ways of finding approximate numerical solutions. In this chapter we will focus on implicit variables that are vectors (not maps) and that can be approximated with simple methods such as Picard iterations. We will consider these implicit variables to be constraints of optimization problems. We call these new class of optimization models, implicit optimization, which includes deep learning, nonlinear control, and mixed-integer programming as special cases. Implicit optimization provides a unified perspective on these different fields, leading to new algorithms and surprising connections. Among others, we will propose a new heuristic to tackle combinatorial problems different from Hopfield Methods. When it comes to solving these type of problems, we propose two type of algorithms: (i) implicit gradient descent and (ii) the Fenchel alternative direction method of multipliers. We will illustrate our theory and methods with a study case in energy systems and numerical experiments on feedforward neural networks and combinatorial optimization problems.

6.1 Introduction

Problem Formulation

We define an implicit optimization problem to be one of the form

$$\begin{aligned} &\min_{x\in\mathcal{X},u\in\mathcal{U}} f(x,u) \\ &x = \phi(x,u) \end{aligned} \tag{$\mathcal{P}$}$$

where the objective function f is convex, and $\mathcal{X}$ and $\mathcal{U}$ are convex subsets of $\mathbb{R}^n$ and $\mathbb{R}^q$ respectively. The map $\phi : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$, assumed to be continuous and differentiable almost everywhere, is referred to as the *implicit map*, while the equation $x = \phi(x, u)$ is called the *implicit equation*. We use the terminology *implicit* because the optimization variable x is defined implicitly – meaning there is no analytical formula for it – as a solution to the implicit equation. We call x the *implicit state* and u the *control input*. We will consider two possible assumptions for the implicit map ϕ: the *contractive mapping* and the *Fenchel* conditions, presented in Section 6.2.

Related work

Implicit operators in convex optimization methods are ubiquituous: they arise in proximal gradient [153], the alternative direction method of multipliers (ADMM) [27] conjugate gradients and mirror descents [20], to cite some examples.

In this chapter we consider the following questions: What are the well-known problems that can be cast in that framework? How can they be generalized? And finally, what are some methods that would allow to tackle these type of problems?

We are not the first to consider optimization variables that satisfy a form of implicit constraints. Ordinary differential equations (ODEs) or partial differential equations (PDEs) define the state trajectories in an implicit way. Therefore optimization over ODEs and PDEs [92] is a form of implicit optimization.

Complementarity problems can also be seen as a form of implicit optimization [39]. These type of problems typically arise when an optimization problem has an optimization variable x that is constrained to be the solution to an optimization problem itself. This is a modeling approach that is often considered in energy market applications [53]. Implict equations arise in deep learning as well. In [47] the authors have proposed a new framework where the hidden sates are defined in an implicit manner. In the same area, recent works have also considered implicit constraints. For example, [84, 14] show how to use an implicit framework for the task of sequence modeling. In [35] the authors use implicit constraints to solve and construct a general class of models known as neural ordinary differential equations. Finally, in [10] the authors tackle the fundamental problem of selection bias in machine learning with the so-called invariant risk minimization that also uses implicit constraints.

Contribution and chapter outline

In this chapter we provide a unified perspective on implicit optimization models. We focus on two possible assumptions for the implicit map: the contraction condition and the Fenchel condition. We show that these two conditions allow one to encompass a multitude of optimization problems across different fields. We also propose novel algorithms based on these formulations, as well as extension of these models.

In Section 6.2 we present the two aforementioned possible conditions on the implicit map and exhibit some of their mathematical properties. In particular, we show that the Fenchel condition is satisfied, up to an approximation, under some mild conditions.

In Section 6.3 we give examples of known optimization problems that can be cast as implicit optimization models and explore the structure of their corresponding implicit maps.

In Section 6.4 we provide some generic methods for solving implicit optimization models and in Section 6.5 we put these methods in practice.

In this preliminary work, our focus is on theoretical and algorithmic underpinnings, and not on empirical validation. In particular, we do not aim at empirically proving the superiority of our methods with state-of-the-art methods for Deep Learning, MINLP, or nonlinear control with large, real-world problems and datasets. Section 6.5 provides a few examples supporting the theory put forth in this chapter.

6.2 Precursors and mathematical background

In this section, we present the mathematical background for implicit optimization. We start with the contractive mapping conditions and then explore some of its implications. We then present the Fenchel condition and introduce the Fenchel divergence that generalized euclidean L2 penalties. We then show that the Fenchel condition can always be satisfied with some approximations.

Contractive mapping condition

We say that the implicit map ϕ is an r-contraction, $r \in [0, 1)$, if for all $u \in \mathcal{U}$,

$$\|\phi(x_1, u) - \phi(x_2, u))\| \leq r\|x_1 - x_2\|, \quad \forall x_1, x_2 \in \mathbb{R}^n$$

If ϕ is an r-contraction, as a direct consequence of the Banach fixed-point theorem [63], given $u \in \mathcal{U}$, x is uniquely defined by the implicit equation. In other words, under the r-contraction assumption, x is a function of u. In that case we will consider $x \in \mathbb{R}^n$ and disregard $\mathcal{X}$, since x is fixed given u. Given u, we can compute $x(u)$ using Picard iterations

$$x^{k+1}(u) = \phi(x^k(u), u)$$

starting with any $x^0(u) \in \mathbb{R}^n$. Picard iterations converge linearly to $x(u)$ and,

$$\|x^k(u) - x(u)\| \leq r^k\|x(u) - x^0(u)\|$$

as shown in [20], proposition A.26. The following theorem allows us to understand the continuity properties of the map $u \to x(u)$.

Theorem 6.2.1. *If the implicit map ϕ is p times continuously differentiable then so is the map $u \to x(u)$ and*

$$\nabla_u x(u) = (I - \nabla_x \phi(x, u))^{-1} \nabla_u \phi(x, u)$$

Proof. This theorem can be seen as a direct corollary of the implicit function theorem as stated in Proposition A.25 [20] and we only need to prove the non-singularity of $I - \nabla_x \phi(x, u)$. As the map ϕ is a r-contraction we have that $\|\nabla_x \phi(x, u)\|_{op} \leq r$ which proves that none of its eigenvalues are of modulus 1, which proves non-singularity of $I - \nabla_x \phi(x, u)$. □ □

Fenchel condition

We start with the following definition of a convex potential gradient.

Definition 6.2.1. *We say that a vector map $v : \mathcal{D}_v \to \mathbb{R}^p$ with a convex domain $\mathcal{D}_v \subset \mathbb{R}^p$ is a convex potential gradient if there exists a convex and differentiable real-valued function $V : \mathcal{D}_v \to \mathbb{R}$ such that $\nabla V(x) = v(x)$. V is referred to as the potential of v. We denote in short $v \in \mathcal{CPG}$ or say that v is a $\mathcal{CPG}$.*

We say that the implicit map ϕ satisfies the *Fenchel condition* if there exists a matrix A, a bilinear map $\zeta : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ and a map $\psi \in \mathcal{CPG}$ such that

$$\phi(x, u) = A\psi(\zeta(x, u)) \tag{6.1}$$

As a special case, we say that ϕ satisfies the *direct Fenchel condition* if $A = I$. This condition is not the only one that allows the construction of tractable algorithms, as shown in Section 6.3 with the example of MINLP. The reason why such a choice is made for this condition will be made clearer in section 6.4 and once the Fenchel divergence is introduced.

Convex potential gradients

We give examples of $\mathcal{CPG}$ functions and introduce the Fenchel divergence.

Real maps: All real continuous maps have an integral function, therefore ϕ is $\mathcal{CPG}$ if and only if ϕ is non-decreasing and the domain of definition of ϕ is connected. For instance, $x \to \log(x)$ defined on $(0, +\infty)$, $x \to (x)_+$ (ReLU function), $x \to (1/2)x_+^2$ and $x \to \tanh(x)$ defined on $\mathbb{R}$ are $\mathcal{CPG}$s.

Multivariate maps: If ϕ is linear with $\phi(x) = Ax + b$ and a convex domain of definition $\mathcal{X}$, then ϕ is a $\mathcal{CPG}$ if and only if $A \succeq 0$. In that case a corresponding convex potential is given by

$$\Phi(x) = (1/2)x^\top Ax + b^\top x$$

If ϕ is the softmax function, often used in machine learning as a smooth approximation of the argmax,

$$\phi(x) = \frac{\exp(x)}{\left(1^\top \exp(x)\right)}$$

where the exponential is applied elementwise, then ϕ is a $\mathcal{CPG}$ with convex potential $\Phi(x) = \log(1^\top \exp(x))$, which is the LogSumExp, a smooth approximation of the max function.

If $\phi(x) = x/\|x\|_2$ (the Euclidean normalization), with domain of definition $\mathbb{R}_+\backslash\{0\}^n$, then the convex potential is given by $\Phi(x) = \|x\|_2$.

We finish this section by considering the implicit map $\phi(x, u) = x + \nabla_x g(x, u)$ with g convex in x, we directly have that ϕ satisfies the direct Fenchel condition as $\Phi(x, u) = g(x, u) + (1/2)\|x\|^2$ is (strongly) convex in x and $\nabla_x \Phi(x, u) = \phi(x, u)$. Note that the implicit equation here is equivalent to

$$x \in \operatorname{argmin}_{x \in \mathbb{R}^n} g(x, u)$$

Therefore all argmin of convex functions of x satisfy the Fenchel condition. This type of implicit equation is used for the purpose of performing invariant risk minimization in [10] and in deep learning in [11]. This property shows the generality of the Fenchel condition. We will show in Section 6.2 that all implicit equations actually satisfy the Fenchel condition to an approximation under mild conditions.

Fenchel divergence

Given $v \in \mathcal{CPG}$ defined on the convex set $\mathcal{X}$, with a convex potential V. Recall the definition of the Fenchel conjugate of V,

$$V^\star(y) = \max_{x \in \mathcal{X}} x^\top y - V(x)$$

Let $\mathcal{X}^\star$ be the domain of definition of $V^\star$ (a convex subset of $\mathbb{R}^n$ by construction, and we can show that $\mathcal{X}^\star$ is the image of $\mathcal{X}$ through v). For $x \in \mathcal{X}$ and $y \in \mathcal{X}^\star$, we define the Fenchel Divergence as,

$$\mathcal{F}_v(y, x) = V^\star(y) + V(x) - x^\top y$$

Theorem 6.2.2. *The Fenchel Divergence is bi-convex, positive, and equal to zero for a pair (x, y) if and only if $y = v(x)$.*

Proof. The Fenchel conjugate of a function is always convex as the point-wise maximum of linear functions. Given $x = x_0$, $-x_0^\top y$ is linear in y hence $\mathcal{F}_v(x_0, y)$ is convex in y. We can show the same by fixing $y = y_0$.
The Fenchel divergence is positive by construction as

$$\mathcal{F}_v(y, z) = \left(\max_{\tilde{x} \in \mathcal{X}} \tilde{x}^\top y - V(\tilde{x})\right) - \left(x^\top y - V(x)\right) \geq 0$$

This inequality is also known as the *Fenchel-Young inequality*. Finally, take the gradient of $V^\star$ with respect to y in its definition, set to zero, and this directly gives $y = v(x)$. □ □

The main appeal of Fenchel divergence is the ability to associate a penalty function to the constraint $y = v(x)$ that uses the specific structure of the nonlinear constraint. Notably, this allows one to associate a bi-convex penalty function. This is not the case for the L2 penalty, $\|y - v(x)\|_2^2$ which is in general not bi-convex. If $v(x) = x$ (the identity), then a convex potential is $V(x) = (1/2)\|x\|_2^2$ and $V^*(y) = V(y)$. Hence the Fenchel divergence is $\mathcal{F}_{I_d}(x, y) = (1/2)\|x - y\|^2$, which is the classic $L2$ penalty. In that sense, Fenchel divergence is a generalization of the $L2$ penalty, like Bregman divergence is a generalization of the L2 penalty as used in mirror descent.

Fenchel and contraction conditions

Fenchel conditions do not impose unicity of x with respect to the input u. That is, for each input, there might exist a set of solutions $\mathcal{X}(u) := \{x \in \mathcal{X} | x = \phi(x, u)\}$ with strictly more than one element. Generally $\mathcal{X}(u)$ is not convex nor connected. Note that a map ϕ can satisfy both the contractive and Fenchel condition. For instance, suppose ϕ is written as in equation (6.1) with L_ψ-Lipschitz continuous ψ. If $\|A\|_{op}\|B\|_{op} \leq r/L_\psi$, then ψ is an r-contraction. In the following section, we discuss how the Fenchel assumption is general in the sense that we can approximate any implicit map ψ and corresponding implicit states to a precision ε.

Universal approximation theorem

We start by stating a result from [30, 40].

Theorem 6.2.3. *Consider the implicit map $\phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Let us assume $\mathcal{X}$ and $\mathcal{U}$ to be compact subsets of $\mathbb{R}^n$ and $\mathbb{R}^q$ respectively. Let $\psi : \mathbb{R} \to \mathbb{R}$ be a continuous non-constant function. Given a precision ε, there exists an integer $N(\varepsilon)$, matrices $A \in \mathbb{R}^{n \times N(\varepsilon)}$ $B \in \mathbb{R}^{N(\varepsilon) \times n}$ $C \in \mathbb{R}^{N(\varepsilon) \times q}$ and vector $d \in \mathbb{R}^{N(\varepsilon)}$ such that,*

$$\|\phi(x, u) - \hat{\phi}(x, u)\| \leq \varepsilon, \quad \forall x \in \mathcal{X} \text{ and } u \in \mathcal{U} \tag{6.2}$$

where

$$\hat{\phi}(x, u) = A\psi(Bx + Cu + d) \tag{6.3}$$

and ψ is an elementwise nonlinear activation. Moreover, if ϕ is continuously differentiable then we can find ε such that the above property holds as well as

$$\|\nabla\phi(x, u) - \nabla\hat{\phi}(x, u)\| \leq \varepsilon \tag{6.4}$$

In this theorem, the model $\hat{\phi}$ corresponds to a one layer neural network with activation ψ. As we can choose ψ to be the ReLU or tanh functions (for instance), we can approximate any implicit map ϕ defined on a compact set by a map satisfying a special case of the Fenchel condition (6.1) where $\zeta(x, u)$ is simply linear. We still have to show that the use of approximation $\hat{\phi}$ also allows us to approximate the corresponding implicit states.

Approximation under the contraction condition

The approximate implicit equation is given by

$$\hat{x} = \hat{\phi}(\hat{x}, u)$$

In the case where ϕ is a r-contraction and is continuously differentiable, using this theorem, we can find $\hat{\phi}$ such that it is a $(r+\varepsilon)$-contraction (provided we choose $0 < \varepsilon \leq 1 - r$). Hence, for every $u \in \mathcal{U}$, we can guarantee a unique solution $\hat{x}(u)$.

Theorem 6.2.4. *Let $0 < \varepsilon < 1 - r$, let ϕ be a r-contraction defined on a compact set, and let $\hat{\phi}$ be an approximation of ϕ as in equation (6.3) verifying (6.2), (6.4). Given $u \in \mathcal{U}$, let x be the solution to the implicit equation and $\hat{x}$ be the solution to the approximate implicit equation $\hat{x} = \hat{\phi}(\hat{x}, u)$. Then,*

$$\|x - \hat{x}\| \leq \frac{\varepsilon}{1 - r}$$

Proof. We can write $\hat{\phi}(\hat{x}, u) = \phi(\hat{x}, u) + \hat{\varepsilon}$, with $\|\hat{\varepsilon}\| \leq \varepsilon$. Hence,

$$\begin{aligned} \|x - \hat{x}\| &= \|\phi(x, u) - \phi(\hat{x}, u) - \hat{\varepsilon}\| \\ &\leq \|\phi(x, u) - \phi(\hat{x}, u)\| + \varepsilon \\ &\leq r\|x - \hat{x}\| + \varepsilon \end{aligned}$$

therefore $\|x - \hat{x}\| \leq \frac{\varepsilon}{1-r}$ □

□

Approximation in the general case

In the case where ϕ is not a r-contraction and x is not uniquely determined by the input control u, then we can state a similar result provided that a condition on the Jacobian of ϕ holds.

Theorem 6.2.5. *Consider the implicit map ϕ defined on a compact set such that $\forall x \in \mathcal{X}$ and $u \in \mathcal{U}$, the Jacobian matrix $\nabla_x \phi(x, u)$ has none of its eigenvalues equal to one. Given u, let $\mathcal{X}(u)$ be a non-empty set of implicit equation solutions assumed to lie in the interior of $\mathcal{X}$. Then the approximate set of implicit equation solutions $\hat{\mathcal{X}}(u) = \{x \in \mathcal{X} | x = \hat{\phi}(x, u)\}$ is also non-empty and the Hausdorff distance between the two sets is $\mathcal{O}(\varepsilon)$.*

The proof of this theorem is provided in the appendix.

To summarize, from the universal approximation theorem and Theorems 6.2.4 and 6.2.5, we can always consider the implicit map ϕ to satisfy the Fenchel condition (6.1) to an approximation parameterized by ε.

6.3 Applications

In this section we give a non-exhaustive list of classic optimization problems that can be formulated as implicit optimization ($\mathcal{P}$) and show what conditions the corresponding implicit maps satisfy.

Mixed integer programming

Let us start by focusing on the binary constraint $x \in \{0, 1\}$. Interestingly, this is equivalent to $x = x^2$. Generally speaking if we constrain $x \in [K]$, K an integer, this is equivalent to $\prod_{i=0}^{K}(x - i) = 0$. Therefore, we can cast any mixed integer problem in the implicit optimization form with no control input u by taking $\phi(x) = x + \prod_{i=0}^{K}(x - i)$. Let us focus on the combinatorial case $x \in \{0, 1\}$. We also have that $x \in \{0, 1\}$ if and only if $x = \phi(x) = x_+^2$. With that formulation ϕ is $\mathcal{CPG}$ and ϕ satisfies the direct Fenchel condition, as shown in Section 6.2.

An alternative reformulation of $x \in \{0, 1\}$ is

$$x \in [0, 1] \text{ and } \exists z \in [0, 1] \mid x = xz \text{ and } z = xz$$

which given z or x leads to linear equations (that satisfy the Fenchel condition). We can generalize this to higher dimensions with $x \in \{0, 1\}^n$ if

$$x \in [0, 1]^n \text{ and } \exists z \in [0, 1]^n \mid x = x \odot z \text{ and } z = x \odot z$$

We will take advantage of this symmetry and structure in an example in Section 6.5.

Deep Learning

Let us start by considering a fully-connected neural network with L layers that can be written as the sequence

$$x_l = \sigma_l(W_l x_{l-1} + w_l) \tag{6.5}$$

with $l \in [L]$ and where $\sigma_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ is the activation function (assumed to be Lipschitz continuous), $x_l \in \mathbb{R}^{n_l}$ denotes the l-th hidden state, $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and $w_l \in \mathbb{R}^{n_l}$ respectively denote the weight and bias for the layer. We use the compact notation u to refer to the weights and bias and x to refer to the states. The input to the network is denoted $v = x_0 \in \mathbb{R}^{n_0}$. The output and prediction of the network is $v = x^{L+1}$. Let N denote the number of datapoints tuple $\{v(i), y(i)\}$ with $i \in [N]$. The learning objective reads

$$\begin{aligned} \min_{x,u} f(x, u) &:= \sum_{i=1}^{N} \ell(y(i), x_{L+1}(i)) + \mathcal{R}(u) \\ \text{s.to : } & x_l(i) = \sigma_l(W_l x_{l-1}(i) + w_l), \\ & \forall i \in [N] \text{ and } l \in [L + 1] \end{aligned}$$

where ℓ is a convex loss in the second argument, $\mathcal{R}$ is a convex weight regularization term, typically $\mathcal{R}(u) = \rho \sum_{l=1}^{L+1} \left\{ \|W_l\|_F + \|b_l\|_2 \right\}$ (known as L2 regularization), where $\rho > 0$ is a hyperparameter.

We can reformulate the feedforward equations (6.5) as an implicit constraint by noting that it is equivalent to

$$x = \sigma(Wx + w + \tilde{v})$$

with $x \in \mathbb{R}^n$, $n = \sum_{l=0}^{L+1} n_l$ and $x = [x_{L+1}; \cdots, x_0]$, $W \in \mathbb{R}^{n \times n}$ strictly upper-triangular with,

$$W = \begin{bmatrix} 0 & W_{L+1} & & 0 \\ & \ddots & \ddots & \\ & & 0 & W_1 \\ 0 & & & 0 \end{bmatrix}$$

and $w \in \mathbb{R}^n$ with $w = [b_{L+1}; \cdots, b_1; 0]$ and $\tilde{v} = [0; \cdots; v] \in \mathbb{R}^n$. Finally the activation ϕ is defined by block, $\sigma = [\sigma_{L+1}; \cdots; \sigma_1; I_d]$. All in all, we are able to write the learning problem for feedforward neural networks as an implicit problem with implicit equation

$$x = \phi(x, u) = \sigma(Wx + w + \tilde{v})$$

with $u = (W, w)$ and $\tilde{v}$ a constant. Let us highlight the fact that there is an implicit equation and an implicit state defined for each datapoint with a shared controlled input u. It is straightforward to show that the implicit equation verifies the contractive mapping condition. For $x_1, x_2 \in \mathbb{R}^n$,

$$\|\phi(x_1, u) - \phi(x_2, u)\|_2 \leq L_\sigma \|W\|_{op} \|x_1 - x_2\|$$

where L_σ denotes the Lipschitz constant for σ. Since W is strictly upper triangular, $\|W\|_{op} = 0$. Hence, ϕ is a 0-contraction. This result is not surprising since the feedforward neural equation (6.5) is explicit. This actually leads to a natural extension of classical neural networks – what if we do not impose an upper triangular structure on W? This extension is called implicit deep learning and is presented in the next chapter. Namely, the implicit model reads

$$x = \sigma(W_x x + W_v v)$$

with prediction $\hat{y}(v) = \hat{W}_x x + \hat{W}_v v$. with $\sigma \in \mathcal{CPG}$. For notation clarity we omitted the bias terms. Going back to the implicit optimization framework, here the implicit state is x and the control input is $u = [\hat{w}_x, \hat{W}_u, W_x, W_u]$. The implicit model satisfies the direct Fenchel condition as σ is $\mathcal{CPG}$ and $W_x x$ is a bilinear map in (x, W_x). Since we want the prediction rule to be unique with respect to v, we impose $\|W_x\|_{op} < 1/L_\sigma$ which, generally speaking, is not a tractable constraint to impose on W_x. Instead we can choose a tighter convex constraint $\|W_x\|_\infty < 1/L_\sigma$, which is equivalent to n L1 ball constraints of radius $1/L_\sigma$. With that choice we still guarantee the implicit map to be contractive as shown in [47].

Since the matrices W_x and W_u are allowed to be dense, the implicit deep learning model has more representational capability than classic feedforward neural networks. In [47], the authors show that implicit deep learning actually contains as a special case most known deep learning architectures, e.g. convolution neural networks and recurrent neural networks. In Section 6.4 we will derive an implicit gradient method for contractive implicit maps. In the case of Deep Learning this can be seen as an extension of backpropagation to broader deep learning models.

Nonlinear control

Consider a non-autonomous system given by: $\frac{dx}{dt}(t) = g(x(t), u(t))$ with $x(0) = x_0$, and g continuously Lipschitz with respect to x (which ensures existence and uniqueness of the trajectory $x(t)$ from Picard–Lindelöf theorem [136]). Note that, although not directly similar to the implicit equation in $(\mathcal{P})$, an ODE is also an implicit equation in x.

The goal in optimal control is to minimize the objective $\int_0^T c(t, x(t), u(t))dt$ (where c represents a cost function and T a time horizon) while ensuring $x(t) \in \mathcal{X}$ and $u(t) \in \mathcal{U}$. The problem is discretized in time with step-size Δt. We then denote $x = [x(1); \cdots ; x(T)]$, $u = [u(1); \cdots ; u(T)]$ and the objective is given by,

$$f(x, u) = \sum_{t=1}^{T} c_t\big(x(t), u(t)\big)$$

with $c_t(x, u) = c(t, x, u)\Delta t$. There are many ways to discretize the ODE models. A classical method is the forward Euler method,

$$x(t + 1) = x(t) + g\big(x(t), u(t)\big)\Delta t \tag{6.6}$$

This discretization is explicit, and similar to feedforward neural networks, we can rewrite this system in implicit equation form

$$x = \phi(x, u) = Dx + g(Dx, u)\Delta t$$

with D as the strictly upper triangular matrix

$$D = \begin{bmatrix} 0 & I & & 0 \\ & & \ddots & \\ & & & I \\ 0 & & & 0 \end{bmatrix}$$

Similar to the feedforward neural network, ϕ as defined above is 0-contractive, because (6.6) is explicit.

Consider the backward Euler method. Although it is used less in practice, the method is an unconditionally stable numerical schemes and is defined in an implicit manner,

$$x(t) = x(t-1) + g(x(t), u(t))\Delta t$$

Note that $x(t)$ is uniquely defined for $\Delta t < \frac{1}{L_g}$. The corresponding implicit equation reads

$$x = \phi(x, u) = Dx + g(x, u)\Delta t$$

and ϕ is a $L_g\Delta t$-contraction. Consequently, backward Euler method is well-suited for implicit optimization methods.

6.4 Methods

This section introduces two classes of numerical methods to solve implicit optimization programs. In Section 6.5 we will use these methods and their variations to exploit the structure of the implicit optimization problem $(\mathcal{P})$ and implicit map ϕ.

Implicit gradient

As shown in Section 6.2, x is a function of u. Therefore we can reformulate $(\mathcal{P})$ without x,

$$\min_{u \in \mathcal{U}} F(u)$$

with $F(u) = f(x(u), u)$. We can now use classic methods for nonlinear constrained optimization. From Theorem 6.2.1, since $x(u)$ is differentiable w.r.t. u, then $F(u)$ and its gradient is given by,

$$\nabla_u F(u) = \nabla_u f(x, u)|_{x=x(u)} + \nabla_x f(x, u))|_{x=x(u)} \nabla_u x(u)$$

A direct method to solve $(\mathcal{P})$ is to apply projected gradient descent,

$$u^{k+1} = \mathrm{Proj}_{\mathcal{U}}\Big(u^k - \alpha^k \nabla_u F(u^k)\Big)$$

We refer to the reader to the first chapter [20] for judicious choices of step-size. The worst case rate of convergence to a local minimum is then $\mathcal{O}(1/k)$ if F has Lipschitz gradient.

Fenchel alternative direction method of multipliers

Let us assume ϕ satisfies the Fenchel condition. Then, the implicit equation is equivalent to,

$$x = Az \text{ and } z = \psi(\zeta(x, u)), \quad \text{with } z \in \mathbb{R}^n$$

Using the Fenchel Divergence and Thm 6.2.2, this is equivalent to,

$$x = Az \text{ and } \mathcal{F}_\psi(z, \zeta(x, u)) = 0$$

Let us formulate the *Fenchel augmented Lagrangian*,

$$\mathcal{L}_\rho(x,u,z,\lambda) = f(x,u)+\lambda^\top(x-Az)+\frac{\rho_1}{2}\|x-Az\|^2 ... \\ ... + \rho_2 \mathcal{F}_\psi(z,\zeta(x,u))$$

where $\rho_1,\ \rho_2 > 0$ are hyperparameters. Given a dual variable λ, the Fenchel augmented Lagrangian is 3-block multi-convex in (x,u,z). This naturally leads to a method very similar to the alternative direction method of multipliers (ADMM) that we call *Fenchel-ADMM*, given by the following iterations:

Primal updates:

$$x^{k+1} \in \text{argmin}_{x\in\mathcal{X}} \mathcal{L}_\rho(x,u^k,z^k,\lambda^k)$$
$$u^{k+1} \in \text{argmin}_{u\in\mathcal{U}} \mathcal{L}_\rho(x^{k+1},u,z^k,\lambda^k)$$
$$z^{k+1} \in \text{argmin}_{z\in\mathbb{R}^n} \mathcal{L}_\rho(x^{k+1},u^{k+1},z,\lambda^k)$$

Dual update:

$$\lambda^{k+1} = \lambda^k + \rho_1(x^{k+1}-Az^{k+1})$$

Even if the implicit map satisfies the direct Fenchel condition, then introducing an extra variable z may still be necessary, depending on the structure of $\mathcal{F}_\psi(x,\zeta(x,u))$ or $f(x,u)+\rho_2\mathcal{F}_\psi(x,\zeta(x,u))$. Given u, generally speaking, these functions might not be convex in x, which might make the primal updates non-convex programs and therefore potentially not tractable. Nevertheless, in the case of using Fenchel-ADMM for feedforward neural networks, we will show that we can exploit the problem structure without introducing an additional variable z. If no variable z is introduced, then there is no need for a dual variable λ and there is no dual step. In that case, the algorithm consists in the above primal updates only. We call this type of algorithm Fenchel block coordinate descent (Fenchel BCD). From [168], if the objective f is smooth, then we have a guarantee that Fenchel BCD will converge to a Nash equilibrium with a worst case rate of convergence $\mathcal{O}(1/k)$.

There are many alternatives to the algorithms presented above. For instance, instead of doing full-block primal updates, we can take some gradient or proximal steps. Instead of taking ρ_1 as a dual step-size we can choose another step-size rule. We will use such variations in the numerical examples.

6.5 Numerical experiments

In all the following experiments we use Matlab R2019a on a single 2.2 GHz processor.

Mixed integer programming

Let us consider the following combinatorial quadratic program,

$$\min_{x\in\{0,1\}^n} f(x) = \frac{1}{2}x^\top Qx + q^\top x$$

with $Q \succeq 0$. We reformulate this problem equivalently, using the ideas presented in Section 6.3,

$$\min_{x\in[0,1]^n, z\in[0,1]^n} f(x) + f(z)$$
$$x = x \odot z$$
$$z = x \odot z$$

The corresponding augmented Lagrangian is

$$\mathcal{L}_\rho(x,z,\lambda_1,\lambda_2) = f(x) + f(z) + ...$$
$$\lambda_1^\top(x - x\odot z) + \lambda_2^\top(z - x\odot z) + \frac{\rho}{2}\|x-z\|^2$$

where $\rho > 0$. We added the constraint $x = z$ in the penalty for robustness. Note this formulation is symmetric in x and z. We use an algorithm similar to ADMM that reads,

$$x^{k+1} = \text{argmin}_{x\in[0,1]^n} f(x) + (x - z^k \odot x)^\top \lambda_1^k - ...$$
$$(z^k \odot x)^\top \lambda_2^k + \frac{\rho}{2}\|x - z^k\|^2$$

$$z^{k+1} = \text{argmin}_{z\in[0,1]^n} f(z) + (z - x^{k+1} \odot z)^\top \lambda_2^k - ...$$
$$(x^{k+1} \odot z)^\top \lambda_1^k + \frac{\rho}{2}\|z - x^{k+1}\|^2$$
$$\lambda_1^{k+1} = \lambda_1^k + \alpha^k(x^{k+1} - x^{k+1} \odot z^{k+1})$$
$$\lambda_2^{k+1} = \lambda_2^k + \alpha^k(z^{k+1} - x^{k+1} \odot z^{k+1})$$

where α^k is a dual step-size. In the following simulations we will choose a constant step-size $\alpha^k = 0.2$. We can interpret this algorithm as a symmetric back and forth optimization between x and z until a consensus is reached.

As a baseline comparison, we use the CPLEX optimizer `cplexmiqp` [145] as a reference. We start by displaying an example with $n = 40$ and random choices for matrix Q and vector q. It took over 7 minutes for CPLEX to find a solution. We run 100 dual steps and use the Matlab built-in `quadprog` function[154] to do the x and z updates above. Since x and z play symmetric roles, we choose to validate results with one of the optimization variables: x. All in all, it took approximately 1 second for our algorithm to run. In terms of objective function value, we obtain a relative difference of 0.4% between the two methods. Let us define the distance to $\{0,1\}$ to be $d_{\{0,1\}}(x) = \frac{1}{n}\sum_{i=1}^n \min\{|x_i|, |x_i - 1|\}$. We get 0 for CPLEX and 10^{-4} with our method. In this particular instance, at least, our method is a good heuristic for finding an approximate solution to this combinatorial problem. For a better comparison we draw 500 random matrices Q and vector q and solve it using our method and CPLEX. We choose $n = 20$ because CPLEX takes too long too converge (the complexity being exponential in n), although our method seems to perform better for larger n. The following table summarizes the results.

Table 6.1: Comparison of CPLEX and Implicit Optimization

	CPLEX	Implicit Optimization
$\hat{\mathbb{E}}[f(x)]$	-4.82	-4.7
$\hat{\sigma}[f(x)]$	1.94	1.96
$\hat{\mathbb{E}}[d_{\{0,1\}^n}(x)]$	10^{-15}	1.9×10^{-6}
$\hat{\sigma}[d_{\{0,1\}^n}(x)]$	10^{-14}	2.2×10^{-5}

Our method yields solutions that are close to binary, and almost produces the same objective value, on average, relative to an exact solver like CPLEX. This method can be seen as a heuristic method for solving mixed integer nonlinear programs, such as Hopfield neural networks [69, 152] or semi-definite relaxation [121].

Feedforward Neural Networks

Let us consider a two-layer $10 \times 20 \times 15 \times 5$ fully-connected neural network with ReLu activation (as presented in Section 6.3). Here the input is $v = x_0 \in \mathbb{R}^{10}$ and the output is $y = x_3 \in \mathbb{R}^5$. We create a synthetic dataset with $N = 200$ datapoints by randomly drawing the "true" weights of the network (considering no bias terms) and applying the feedforward rule to random inputs. The goal of a learning algorithm is to approximate this "true" network on the training dataset via regression. As previously mentioned, back-propagation algorithms are stochastic versions of the implicit gradient method presented in Section 6.4. As shown in Section 6.3, a feedforward neural network also satisfies the direct Fenchel condition. Using this fact, we develop a Fenchel ADMM algorithm tailored for this type of problem. Let us use the compact matrix notation, $Y = [y(1), \cdots, y(N)]$, and $X_l = [x_l(1), \cdots, x_l(N)]$. We can write the learning problem as,

$$\min_{W_1, W_2, W_3, X_1, X_2} \frac{1}{2N} \|Y - W_3 X_2\|_F^2 + \rho \mathcal{F}_\sigma(X_2, W_2 X_1) \\ ... + \rho^2 \mathcal{F}_\sigma(X_1, W_1 X_0)$$

The Fenchel divergence term for $l \in \{1, 2\}$ is given by,

$$\mathcal{F}_\sigma(X_l, W_l X_{l-1}) = \\ \frac{1}{2N} \Big(\|X_l\|_F^2 + \|(W_l X_{l-1})_+\|_F^2 - 2 \text{ trace}(X_l^\top W_l X_{l-1}) \Big)$$

The way we formulate this problem corresponds to unfolding the neural network in the sense that we consider that the hidden states are also optimization variables. Here the choice of

$\{\rho, \rho^2\}$ can be interpreted as a form of backward penalization. We take ρ to be typically small, here $\rho = 10^{-2}$.

Contrary to Section 6.4 we do not introduce an extra variable z to ensure 3-block multi-convexity. We can see that the objective of the learning problem is already 3-block multiconvex in (W_3, W_2, W_1), X_2 and X_1. Here we use the Fenchel BCD presented in Section 6.4 where we only make a single gradient step for each block iteration. We run 5000 iterations and illustrate the resulting RMSE across iterations in Fig. 6.1. We get a final RMSE of 5.1 which is very close to what can be obtained by using the Deep Learning toolbox on Matlab with the ADAM algorithm.

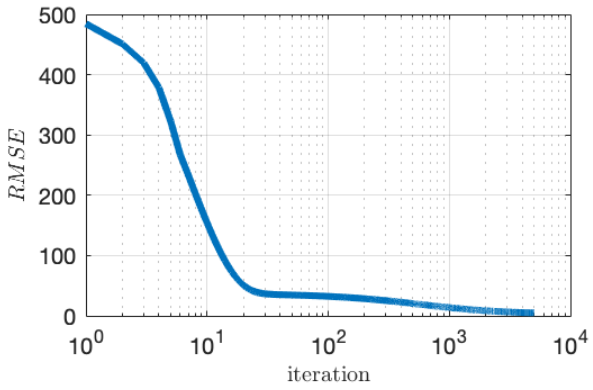

Figure 6.1: RMSE across iterations

Solving overstay of plug-in electric vehicle charging with behavioral modeling

In recent years, the PEV market has been expanding at a faster pace than its supporting infrastructure. Given the infrastructure in place, there is room to improve charging service accessibility via operation. Moreover, we mentioned the challenges facing renewable integration in chapter 2: similarly, PEVs charging schedules can be optimized at a station to alleviate the duck curve. Nevertheless, in this section will not consider the electricity markets but the time-of-use tariffs (TOUs). We consider that the charging station can also serve as a parking spot. Upon arrival, drivers input the desired parking duration and are asked to make a choice: (i)leave, (i) allow a flexible charging schedule, or (iii) charge as fast as possible. If a driver chooses that last option, he will incur an overstay fee for the time the PEV stays idle (i.e. after his vehicle is fully charged - irrespective of the parking duration). The driver that chooses the flexible option will only get charged for overstay after its parking time limit has been reached. Finally, each driver have a third option, choose not to charge and not use the

parking spot (this might be the case if the price is too high). We will define an opportunity cost associated to that choice.

In order to model how a driver makes a decision, we use a discrete choice model [141]: we associate to the two alternatives a utility function of the form

$$V_i(x) = \beta_i^\top x + v_i, \quad i \in \{1, 2, 3\}$$

Where $x \in \mathbb{R}^3$ is a variable that consists in three tariffs: x_1 is the charging tariff for charging with flexibility (i.e. price per kWh), x_2 is the the charging tariff for charging as fast as possible and x_3 is the overstay penalty (i.e. price to be paid per hour for overstaying). The parameter β_i is a vector that is fitted with behavioral observations, and v_i is a constant utility associated with a given choice, regardless of tariffs. Based on the multinomial logit model, the probability p_i of choosing $i \in \{1, 2, 3\}$ is given by

$$p_i := \phi(Bx + v)_i = \Big[\frac{e^{Bx+c}}{1^\top e^{Bx+v}}\Big]_i$$

Where ϕ is the softmax function already introduced earlier in the chapter. Here B is the matrix of stack vector parameters $[\beta_1^\top; \beta_2^\top; \beta_3^\top]$, similarly $v = [v_1; v_2; v_3]$. The objective we consider is to minimize the overall cost associated with operation of the charging station given by

$$f(x) = \sum_{i \in \{1,2,3\}} p_i h_i(x) = \phi(Bx + v)^\top h(x)$$

We now define the cost associated to each choice $i \in \{1, 2, 3\}$.

First, we define h_1 to be the cost associated to flexibility and given by

$$h_1(x) = \min_{u \in \mathcal{U}} h_1(x, u) := \lambda \frac{T\ \overline{x}_3}{x_3} + u^\top (c - x_1) + \frac{\rho}{2}\|u\|^2$$

where T is the average duration of overstay without any overstay penalty x_3 and $\overline{x}_3$ is a baseline overstay tariff, λ is a regularization parameter (homogeneous to a cost per unit of time). The variable u corresponds to the charging decision variables (its components are the charging power in kW, and component indices represent time). We denote by c the time of use cost (i.e. the price per unit of power that has to be paid to the utility grid for a given time period). The set $\mathcal{U}$ is convex and includes the dynamics and objective associated with charging PEVs, much like in chapters 2, 3, 4. The reader will have noted that the cost function h_1 is an optimization problem: it corresponds to the optimal operation of the station given the charging tariff x_1 (that is constant with respect to time contrary).

Second, we define h_1 to be the cost associated to the choice of charging as fast as possible

$$h_2(x) = \lambda \frac{T\ \overline{x}_3}{x_3} + \sum_{t=1}^{T_f} (c_t - x_2)\overline{u}$$

where T_f is the time needed to charge fully given the initial state of charge, and $\overline{u}$ is the maximum charging power that can be provided.

Third, we define h_3 to be the cost associated to a driver choosing not to charge, it corresponds to a lost opportunity cost given by

$$h_3(x) = \sum_{t=1}^{T_f} c_t \overline{u}$$

All in all, the optimization problem for managing the station has the form

$$\min_x \phi(Bx + v)^\top h(x)$$

Which is not a convex problem. Let us It can be reformulated in an implicit form by including u as an optimization variable

$$\begin{aligned} \min_{x,u} \quad & \phi(Bx + v)^\top h(x, u) \\ & u = \operatorname*{argmin}_{u' \in \mathcal{U}} (c - x_1)^\top u' + \frac{\rho}{2}\|u\|^2 \end{aligned}$$

Remark that as mentioned earlier the argmin satisfies the Fenchel condition. Nevertheless we remark that because of the structure of h and the fact that $\phi(\cdot) > 0$ we can rewrite this problem as

$$\min_{x,u \in \mathcal{U}} \phi(Bx + v)^\top h(x, u)$$

Which can be rewritten as

$$\begin{aligned} \min_{w,x,u \in \mathcal{U}} \quad & w^\top h(x, u) \\ & w = \phi(Bx + v) \end{aligned}$$

Where we an explicit constraint that satisfies the Fenchel condition. We can leverage the techniques developed in the previous section to derive an algorithm based on the Fenchel alternative direction method of multipliers. The Fenchel divergence associated with the the soft-max $\phi(\cdot)$ is given by

$$\mathcal{F}_\phi(x, y) = \operatorname{lse}(x) + \operatorname{lse}^\star(y) - x^\top y, \quad x \in \mathbb{R}^n, \text{ with } y \geq 0 \text{ and } 1^\top y = 1$$

where $\operatorname{lse}(\cdot)$ is the log-sum-exp, $\operatorname{lse}(x) = \log(1^\top \exp(x))$, the Fenchel conjugate of which is given by

$$\operatorname{lse}^\star(y) = \begin{cases} y^\top \log(y) \text{ if } y \geq 0 \text{ and } 1^\top y = 1 \\ \infty \text{ otherwise} \end{cases}$$

The algorithm we use reads

$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_x h(x, u^k)^\top w^k + \rho \mathcal{F}_\phi(w^k, Bx + v) \\ u^{k+1} &= \operatorname*{argmin}_{u \in \mathcal{U}} h(x^{k+1}, u)^\top w^k \\ w^{k+1} &= \operatorname*{argmin}_w h(x^{k+1}, u^{k+1})^\top w + \rho \mathcal{F}_\phi(w, Bx^{k+1} + v) \end{aligned}$$

Remark that each update correspond to a convex optimization problem because the function h is bi-convex in (x, u). In the following we succinctly provide some simulation results, we do not go into details as to the choice of tuning parameters and the data used, we refer the reader to [171] for a more in depth understanding. In this case study, we consider data from the charging facilities of Cal Poly campus, which is a workplace charging station for staff and students. This data is used to build a statistical model for arrivals and energy needs. We simulate for 50 days the operation of this facility with a controller using the algorithm we described earlier. As it can be seen in figure 6.2, over this time period, with our method, we decrease by more than 37% the mean overstay duration, we increase by 10% the profit made from the station and we increase the number of vehicles that have been charged (service provided) by 27%.

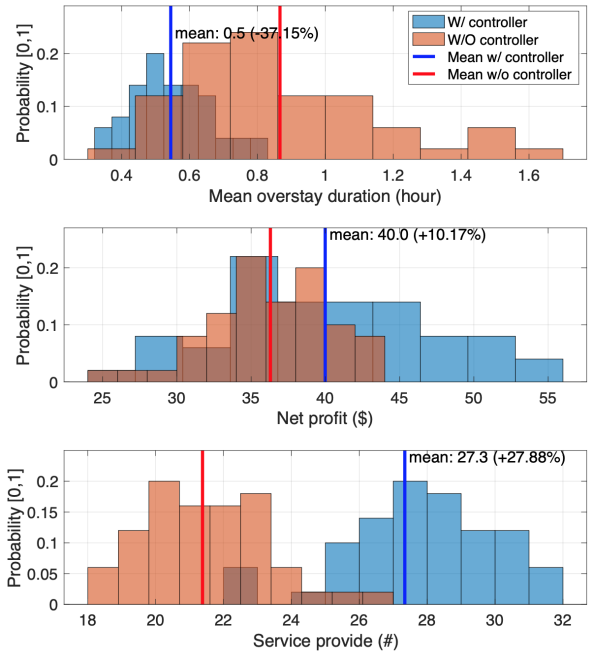

Figure 6.2: Monte Carlo simulation results with a total of 50 days of operations.

Conclusion

We have presented a new class of (non-convex) optimization models called implicit optimization that includes a wide variety of optimizations models. We provided various examples for applications: ranging from nonlinear control to deep learning, and we have shown how these models can be unified into the implicit optimization framework. We believe that implicit

optimization models could include many more optimization models than the ones that have been discussed in this chapter.

6.6 Appendix

Proof of Theorem 6.2.5

Proof. Let us define the (asymmetric) distance between the set $\mathcal{X}(u)$ and $\hat{\mathcal{X}}(u)$,

$$d\big(\mathcal{X}(u); \hat{\mathcal{X}}(u)\big) = \max_{x \in \mathcal{X}(u)} \min_{\hat{x} \in \hat{\mathcal{X}}(u)} \|x - \hat{x}\|$$

The Hausdorff distance between the two sets is defined as,

$$d_H(\mathcal{X}(u), \hat{\mathcal{X}}(u)) = \max \Big\{ d\big(\mathcal{X}(u); \hat{\mathcal{X}}(u)\big), d\big(\mathcal{X}(u); \hat{\mathcal{X}}(u)\big) \Big\}$$

For all $x \in \mathcal{X}$, we write $\hat{\phi}(x, u) = \phi(x, u) + \epsilon(x)$, with $\epsilon(x) \leq \varepsilon$. We assume the map $\epsilon(\cdot)$ is $\varepsilon-$Lipschitz: $\|\nabla \epsilon(x)\| \leq \varepsilon$ (as guaranteed by the universal approximation theorem). Let x be an element of $\mathcal{X}(u)$ and let neighborhood $\mathcal{N} = \mathcal{B}(x, \eta) \subset \mathcal{X}$ be a ball with radius $\eta > 0$ centered at x. Let $(x + \delta) \in \mathcal{N}$, we have

$$\hat{\phi}(x + \delta, u) = \phi(x + \delta, u) + \epsilon(x + \delta)$$

From the mean value theorem, there exists $x' \in [x, x + \delta]$ (and therefore $x' \in \mathcal{N}$ by convexity) such that,

$$\phi(x + \delta, u) = \phi(x, u) + \nabla_x \phi(x', u)\delta$$

Using this equation, and the previous one, and the fact that $x = \phi(x, u)$ gives

$$\hat{\phi}(x + \delta, u) = x + \nabla_x \phi(x', u)\delta + \epsilon(x + \delta)$$

We would like to show that there exists a δ such that $x + \delta = \hat{\phi}(x + \delta, u)$. By hypothesis, as $I - \nabla_x \phi(x', u)$ is non-singular, this is equivalent to

$$\delta = (I - \nabla_x \phi(x', u))^{-1} \epsilon(x + \delta) \tag{6.7}$$

Let us define

$$\Lambda = \max_{x' \in \mathcal{N}} \|(I - \nabla_x \phi(x', u))^{-1}\|$$

which is well defined by compactness of $\mathcal{N}$. Then the function of δ in the right hand side of (6.7) is $\Lambda\varepsilon-$ Lipschitz. If we take $\varepsilon < 1/\Lambda$, then this function of δ is a contraction and there exists a unique δ satisfying (6.7). By synthesis $\delta < \Lambda\varepsilon$ and we can choose ε such that $\Lambda\varepsilon < \eta$. If we choose $\hat{x} = x + \delta$, $\hat{x}$ is in $\hat{\mathcal{X}}(u)$ and at a distance bonded by ε times a constant. A similar symmetric argument allows us to show that if we take $\hat{x} \in \hat{\mathcal{X}}(u)$ we can find a similarly $x \in \mathcal{X}(u)$ such that $\|\hat{x} - x\| = \mathcal{O}(\varepsilon)$. Since ϕ is assumed to be defined on a compact set, then we directly have that the Hausdorff distance is $\mathcal{O}(\varepsilon)$ □

□

Chapter 7

Implicit Deep Learning

Implicit deep learning is a special case of implicit optimization. In this chapter, we will study in more detail how we can generalize the recursive rules of feed forward neural networks. As described in the previous chapter, the prediction rules are based on the solution of a fixed-point equation involving a single vector of hidden features, which is implicitly defined. We will see that this framework greatly simplifies the notation of deep learning. We then derive sufficient conditions ensuring well-posedness of the rule, as well as results pertaining to composition of implicit models. Finally, we derive a stochastic projected gradient method using implicit differentiation to train our models which we benchmark against traditional feedforward networks.

7.1 Introduction

Implicit prediction rules

In this chapter, we introduce Implicit Deep Learning (IDL), an extension of deep learning models. In traditional deep networks, data points are sequentially transformed through different nonlinear layers of a network. In IDL models, the prediction rule is based on solving a fixed-point, "equilibrium" equation in some single *hidden state* vector, specifically:

$$x = \phi(Ax + Bu) \qquad \text{(equilibrium equation)}$$
$$\hat{y} = Cx + Du \qquad \text{(prediction rule)}$$

where $x \in \mathbb{R}^n$ is a vector of "hidden" features, $\phi : \mathbb{R}^n \to \mathbb{R}^n$ is an *activation* (a nonlinear vector map), $u \in \mathbb{R}^p$ is the input data point, $\hat{y} \in \mathbb{R}^q$ is the prediction (or output) and the matrices A, B, C, D are the model parameters. For notational simplicity, the equilibrium equation and prediction rule do not contain any bias terms; we can easily account for these by considering the vector $[u; 1] \in \mathbb{R}^{p+1}$ instead of u, thereby increasing the column dimension of matrices B and D by one. Remark that these notation are different from that of the previous chapter in order to match more closely the notation that is used in Deep Learning.

In the IDL setting, both $\hat{y}$ and x are (implicit) functions of the input vector u, although our notation does not reflect this dependence. We can think of x as a state corresponding to n hidden features extracted from the input based on the equilibrium equation. In general the equilibrium equation may have multiple or no closed-form solution; we discuss the important issue of well-posedness, as well as the problem of solving this equation in Section 7.3.

Perhaps surprisingly, as shown in Section 7.4, the IDL framework includes most current neural network architectures as special cases. IDL models have much more capacity for a given network size (total number of hidden features), as measured by the number of parameters for a given dimension n of the hidden sate.

Contributions and chapter outline

The IDL framework opens up the possibility of novel architectures and prediction rules for deep learning, bypassing any notion of network or layers, as is classically understood. The notational simplicity of IDL models allows one to consider rigorous approaches to challenging problems in deep learning, ranging from robustness analysis, model sparsity and compression, interpretability, and feature selection. Since the IDL framework allows for cycles in the network, we may compose IDL models, for example via feedback connections, which is not permitted under the current paradigm of deep networks.

Reserving to a future publication [8] a full exposition of the potential benefits of IDL, in this chapter we focus on the following contributions:

- *Well-posedness, composition and continuity* (Section 7.3): We establish rigorous and numerically tractable sufficient conditions for the equilibrium equation to be well-posed and solved. Such conditions are then imposed as constraints in the training problem, guaranteeing the well-posedness of a learned prediction rule. We also discuss the composition of implicit models, via cascade or feedback connections for example, in a way that preserves well-posedness; the section ends with a result on the continuity and differentiability of the prediction with respect to the input.

- *Implicit models of neural networks* (Section 7.4): Building on the composition rules of Section 7.3, we provide details on how to represent a wide variety of neural networks as implicit models (including fully connected feed forward networks, convolutional networks, residual networks and recurrent neural networks).

- *Training problem: formulations and algorithms* (Section 7.5): We develop a stochastic projected gradient method for the training problem, leveraging the implicit differentiation technique, and relying on our well-posedness sufficient conditions.

In this preliminary work, our focus is on theoretical and algorithmic underpinnings, and not on empirical validation. In particular, we do not aim at empirically proving the superiority of IDL models over current state-of-the-art deep learning models on large, real-world data sets. Section 7.6 provides a few experiments supporting the theory put forth in this chapter.

7.2 Related Work

In implicit deep learning

Recent works have considered versions of implicit models, and demonstrated their potential in deep learning. The pioneering works by Kazi, Kolter and their collaborators [84, 14, 90] demonstrate the success of an entirely implicit framework for the specific task of sequence modeling (Kazi et al. refer to their models as implicitly-defined neural networks while Kolter et al. refer to their models as Deep Equilibrium Models). [35] use implicit methods to solve and construct a general class of models known as neural ordinary differential equations, while [13] uses implicit models to construct a differentiable physics engine that enables gradient-based learning and high sample efficiency. Several chapters explore the concept of integrating implicit layers with modern deep learning methods, in a variety of ways. For example, [163] show promise in integrating logical structures into deep learning by incorporating a semidefinite programming (SDP) layer into a network in order to solve a (relaxed) MAXSAT problem. In [6], the authors propose to include a model predictive control as a differentiable policy class for deep reinforcement learning, both of which can be interpreted as implicit architectures. In [5] the authors introduced implicit layers where the activation is the solution of some quadratic programming problem; in [44], the authors incorporate stochastic optimization formulation for end-to-end learning task, in which the model is trained by differentiating the solution of a stochastic programming problem.

In lifted models

In implicit learning, there is usually no way to express the state variable in closed-form, which makes the task of computing gradients with respect to (w.r.t.) model parameters challenging. Thus, a natural idea in implicit learning is to keep the state vector as a variable in the training problem, resulting in a higher-dimensional (or, "lifted") expression of the training problem. The idea of lifting the dimension of the training problem in (non-implicit) deep learning by introducing "state" variables has been studied in a variety of works; a non-extensive list includes [148], [11], [66], [170], [173], [31] and [98]. Lifted models are trained using block coordinate descent methods, Alternating Direction Method of Multipliers (ADMM) or iterative, non-gradient based methods. In this work, we introduce a novel aspect of lifted models, namely the possibility of defining a prediction rule implicitly.

Notation

7.3 Well-Posedness, Composition, Continuity

Assumptions on the activation map

We restrict our attention to activation maps ϕ that satisfy the following two conditions:

1. *block-wise:* The map ϕ acts in a block-wise fashion, that is, there exist a partition of n, $n = n_1 + \ldots + n_L$, such that for every vector partitioned into the corresponding blocks: $z = (z_1, \ldots, z_L)$ with $z_l \in \mathbb{R}^{n_l}$, $l \in [L]$, we have $\phi(z) = (\phi_1(z_1), \ldots, \phi_L(z_L))$ for some activations $\phi_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$, $l \in [L]$.

2. *Non-expansive:* For every $l \in [L]$, the maps ϕ_l are Lipschitz-continuous with constants $\gamma_l \leq 1$ with respect to some l_{p_l}-norm, for some integer $p_l \geq 1$:

$$\forall\, u, v \in \mathbb{R}^{n_l} \;:\; \|\phi_l(u) - \phi_l(v)\|_{p_l} \leq \gamma_l \|u - v\|_{p_l}.$$

Our framework can be extended to the case when the Lipschitz constants γ_l, $l \in [L]$, are larger than 1; we imposed non-expansiveness as opposed to the more general Lipschitz continuity for notational simplicity only. In the sequel, we omit the dependence on the underlying activation structure information (i.e. the integers n_l, p_l, $l \in [L]$ and Lipschitz constant γ_l).

Our assumptions encompass most popular activation maps, including ReLU or leaky-ReLU, tanh, sigmoid, each applied componentwise, so that the block-wise assumption holds, with $n_l = 1$, $l \in [L]$. Our model also allows for maps that do not operate componentwise, such as the softmax function, which is indeed non-expansive [54].

Well-posedness

As mentioned in the introduction the IDL model can be ill-posed because the equilibrium equation may have none or multiple solutions. For example, consider the scalar cases $x = \phi(ax + 1)$, with a, x scalars: for $\phi = \tanh(\cdot)$, and $a \geq 3$, there are always three solutions; for $\phi = \max(0, .)$, for $a \geq 1$ there is no solution. This observation leads us to the following definition.

Definition 7.3.1 (Well-posedness)**.** *Matrix $A \in \mathbb{R}^{n \times n}$ is said to be well-posed for a given activation map ϕ (in short, $A \in WP(\phi)$) if, for every $b \in \mathbb{R}^n$, the equilibrium equation in x:*

$$x = \phi(Ax + b) \tag{7.1}$$

has a unique solution.

For example any strictly upper- (or, lower-) triangular matrices are in WP(ϕ), no matter what the map ϕ is. The hidden state x can then be easily obtained via backward (or, forward) substitution. Such triangular matrices arise when modeling Feed Forward Neural Networks (FFNN) with IDL, as seen in Section 7.4.

Tractable sufficient conditions for well-posedness

Our goal now is to derive a numerically tractable (sufficient) condition for well-posedness, and to pave the way for making the learning of IDL parameters tractable as well.

Theorem 7.3.1 (Perron-Frobenius condition for well-posedness)**.** *Given an activation ϕ that satisfies the conditions of Section 7.3, we have $A \in WP(\phi)$ if the Perron-Frobenius eigenvalue of $|A|$ satisfies $\lambda_{pf}(|A|) < 1$. In such a case, for any $b \in \mathbb{R}^n$, the solution to the equilibrium equation (7.1) can be numerically computed via a fixed-point iteration, initializing with $x = 0$ and iterating until convergence:*

$$x \leftarrow \phi(Ax + b). \tag{7.2}$$

The theorem is a rather direct consequence of the global contraction mapping theorem [136]. We provide a complete proof in the Supplementary Materials (SM), §7.7. As shown in the proof, the fixed-point iteration (7.2) has linear convergence, and each iteration is a matrix-vector product, the complexity of which is comparable to that of a forward pass through a neural network of similar size (total number of hidden features).

The condition $\lambda_{pf}(|A|) < 1$ is not convex in A, and maybe difficult to impose in a training problem. The stricter condition $\|A\|_\infty < 1$ is convex and the corresponding projection problem is amenable to efficient algorithms [45, 8]. The following theorem shows that for positively homogeneous activation maps, for which $\phi(\lambda x) = \lambda\phi(x)$ for every x and $\lambda \geq 0$, the two constraints $\lambda_{pf}(|A|) < 1$ and $\|A\|_\infty < 1$ are generically equivalent in the sense that the latter can be used in training problems without loss of generality.

Theorem 7.3.2 (Rescaling IDL models)**.** *Assume that ϕ satisfies the conditions of Section 7.3 and is positively homogeneous. Consider an IDL model with parameters (A, B, C, D) that satisfies the sufficient condition for well-posedness of Theorem 7.3.1, $\lambda_{pf}(|A|) \leq r$, where $r \in [0, 1)$. If the latter is simple, there exists an equivalent IDL model, with the same output y for any given input u, having the same activation ϕ and parameters (A', B', C', D') such that $\|A'\|_\infty < 1$.*

Proof. As seen in [18], the PF eigenvalue of $|A|$ can be represented as

$$\lambda_{pf}(|A|) = \inf_S \ \|SAS^{-1}\|_\infty \ : \ S = \mathrm{diag}(s), \ \ s > 0$$

When the eigenvalue is simple, the optimal scaling vector s is positive: $s > 0$, and the new model matrices are obtained by diagonal scaling,

$$\left(\begin{array}{c|c} A' & B' \\ \hline C' & D' \end{array}\right) = \left(\begin{array}{c|c} SAS^{-1} & SB \\ \hline CS^{-1} & D \end{array}\right),$$

where $S = \mathrm{diag}(s)$, with $s > 0$ a Perron-Frobenius eigenvector of $|A|$. □

Examples of positively homogeneous activations are the ReLU and leaky ReLU. The technique used in the proof of Theorem 7.3.2 allows us to rescale implicit models, such as the ones derived from deep neural networks (see Section 7.4), so that the norm condition $\|A\|_\infty < 1$ is satisfied.

Composition of implicit models

Composition refers to various connections between implicit models, such as cascade, parallel and feedback connections. We first examine cascade connections.

Theorem 7.3.3 (Well-posedness of block-triangular matrices)**.** *Assume that the activation map ϕ acts componentwise on the upper block-triangular matrix*

$$A := \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

then $A \in WP(\phi)$ if and only if the diagonal blocks $A_{11} \in WP(\phi_1)$ and $A_{22} \in WP(\phi_2)$.

Proof. See SM §7.7. □

A similar result holds with lower block-triangular matrices. Now consider two IDL models with matrix parameters (A_i, B_i, C_i, D_i) and activations ϕ_i, $i = 1, 2$. The corresponding *cascaded* model reads

$$\begin{cases} \hat{y}_2 &= C_2 x_2 + D_2 u_2 \\ u_2 &= \hat{y}_1 = C_1 x_1 + D_1 u_1, \\ x_i &= \phi_i(A_i x_i + B_i u_i), \quad i = 1, 2. \end{cases}$$

As shown in [8], the cascaded system is an IDL, for appropriate matrices A, B, C, D, with A block-triangular (details are left to the reader) and a block-wise activation $\phi((z_1, z_2)) = (\phi_1(z_1), \phi_2(z_2))$. Thanks to Theorem 7.3.3, the cascaded system is well-posed if and only if each subsystem is.

To illustrate the result: it is common to have an activation at the output level, for example with a classification task using a softmax at the last layer, resulting in the model

$$x = \phi(Ax + Bu), \quad \hat{y} = \tilde{\phi}(Cx + Du),$$

with $\tilde{\phi}$ the output activation function. We can represent this as an IDL model, by introducing a new state variable; we obtain a cascaded IDL, which is well-posed if and only if $A \in \mathrm{WP}(\phi)$.

Similar composition results hold if we use two (or more) well-posed IDL models in parallel and define the output as either the sum of the outputs $\hat{y} = \hat{y}_1 + \hat{y}_2$, the concatenation $\hat{y} = (\hat{y}_1, \hat{y}_2)$ or affine transformation of it. Furthermore, IDL models are closed under multiplicative and feedback connections, as shown in SM, §7.7. Note that if we connect two neural networks in a feedback fashion, the resulting system is not an ordinary network, but an implicit one.

Continuity and differentiability properties

As shown in Section 7.4, IDL models include as a special case FFNNs. As proved by [104], such models with bounded width are universal approximators (that is, they can approximate any continuous function of $\mathbb{R}^p$); thus, similar properties hold for IDL models. In the case of neural networks, as the rules are explicit, it is easy to show that the continuity property ($\mathcal{C}^d$-continuity, with $d \geq 0$) of the activations leads to the ($\mathcal{C}^d$) continuity of the output w.r.t. the input. We show that this is also true for IDL models.

Theorem 7.3.4. *Consider an IDL model that satisfies the sufficient well-posedness condition $\lambda_{pf}(|A|) < 1$. Then the prediction rule $u \to \hat{y}(u)$ is $\mathcal{C}^d$ if the activation ϕ is $\mathcal{C}^d$. The result also holds if the continuity properties are satisfied a.e.*

Proof. The proof of this theorem is given in SM, §7.7 and relies on the *implicit function theorem* [19] (proposition A.25). □

From the above theorem we have that IDL model with ReLU acivation is continuous and differentiable a.e. w.r.t. u. If $\phi = \tanh$ than the IDL model is $\mathcal{C}^\infty$.

7.4 Implicit models of deep neural networks

A large number of deep neural networks can be modeled as IDL models. Our goal here is to show how to build a well-posed IDL model for a given neural network, with activation maps that satisfy the conditions in Section 7.3. Thanks to the composition rules of Section 7.3, it suffices to model individual layers, since a neural network is nothing more than a cascade of such layers. The block structure (condition 1 in Section 7.3) then emerges naturally as the result of such layer-wise composition. We will show that the corresponding IDL models have strictly (block) upper triangular matrix A, which automatically implies that these models are well-posed, as the equilibrium equation can be simply solved via backwards substitution which corresponds to a simple forward pass through a neural network. Such models also naturally satisfy the PF condition for well-posedness as all the eigenvalues of $|A|$ are zero, since A is then strictly upper-triangular.

Fully connected FFNNs

Consider the following FFNN with $L > 1$ fully connected layers, with a rule of the form $\hat{y} = W_L x_L$, where $x_{l+1} = \phi_l(W_l x_l)$, $l \in [L]$, and $x_0 = u$ the input data point. We can express

this model as an IDL with $x = (x_L, \dots, x_1)$, and

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{ccccc|c} 0 & W_{L-1} & \dots & 0 & 0 \\ & 0 & \ddots & \vdots & \vdots \\ & & \ddots & W_1 & 0 \\ & & & 0 & W_0 \\ \hline W_L & 0 & \dots & 0 & 0 \end{array}\right),$$

and with an appropriately defined block-wise activation function $\phi = (\phi_L, \cdots, \phi_1)$. As already mentioned, due to the strictly upper triangular structure of A, the system satisfies the Perron-Frobenius sufficient condition for well-posedness of Theorem 7.3.1: since $|A|$ is strictly upper-triangular, its Perron-Frobenius eigenvalue is actually zero.

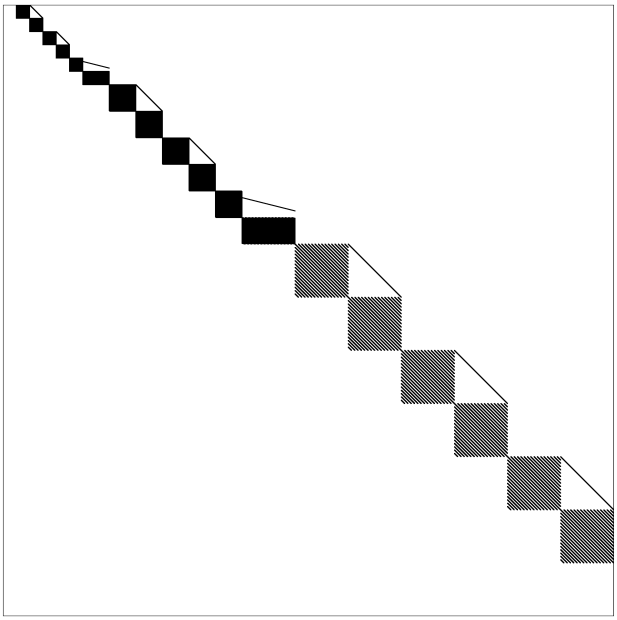

Figure 7.1: Sparsity pattern of the A matrix of a ResNet20 model [73] converted into the IDL framework. The IDL-converted model achieves a test set accuracy of 92.7% on CIFAR-10 and includes shortcut and batch normalization.

Convolutional layers and pooling

A single convolutional layer can be represented as a linear operation $v \to Tv$ where v is an input vector, and T is a matrix representation of the convolution operator, having a Toeplitz-like structure [138]. Hence, we immediately have a way of representing a convolution layer using strictly upper triangular matrices (as in the previous Section 7.4) with Toeplitz block matrices. A convolution layer is often combined with a max-pooling operation. The latter forms a down-sampling operation and has a Lipschitz constant of 1 [158]. Average pooling can also be represented in the IDL framework, since it is a linear operation.

Residual nets

A residual block consists in the following operation,

$$z = \phi_2(v + W_2\phi_1(W_1 v)).$$

The above is a special case of an IDL model: defining the block-wise map $\phi((v_1\ , v_2)) = (\phi_1(v_1), \phi_2(v_2))$, we represent the above as

$$\begin{pmatrix} \tilde{z} \\ z \end{pmatrix} = \phi\left(\begin{pmatrix} 0 & 0 \\ W_2 & 0 \end{pmatrix}\begin{pmatrix} \tilde{z} \\ z \end{pmatrix} + \begin{pmatrix} W_1 \\ I \end{pmatrix} v\right),$$

where $\tilde{z}$ is a new "state" vector. Figure 7.1 displays matrix A for an implicit model equivalent to a 20-layer residual network. Convolutional layers appear as matrix blocks with Toeplitz structure; residual units correspond to the straight lines on top of the blocks. The network only uses the element-wise ReLU activation, except for the last layer, which outputs probability distributions via a softmax.

Recurrent units

Recurrent neural nets (RNNs) can be represented in an *unrolled* form [83]. In a RNN block, the input is a sequence of vectors $\{u_1, \cdots, u_T\}$, where for every $t \in [T]$, $u_t \in \mathbb{R}^p$. At each time step, the network takes in a input u_t and the previous hidden state x_{t-1} to produce the next hidden state x_t; the hidden state x_t defines the *memory* of the network. Precisely, a RNN model reads

$$\begin{aligned} x_t &= \phi_x(W_x x_{t-1} + W_u u_t) \\ y_t &= \phi_y(W_y x_t) \end{aligned}, \quad t \in [T] \tag{7.3}$$

Eqn. (7.3) can be expressed as an IDL model with $x = (x_T, \ldots, x_0)$, $u = (u_T, \ldots, u_1)$, and weight matrices

$$\left(\begin{array}{ccccc|cccc} 0 & W_x & \cdots & 0 & 0 & W_u & \cdots & 0 & 0 \\ 0 & 0 & W_x & \vdots & \vdots & 0 & W_u & \vdots & \vdots \\ & & \ddots & \ddots & 0 & & \ddots & \ddots & 0 \\ & & & 0 & W_x & & & 0 & W_u \\ \hline W_y & 0 & \cdots & 0 & 0 & 0 & \cdots & \cdots & 0 \end{array}\right)$$

Here, A is strictly upper block triangular (the IDL model is therefore well-posed) and has a special structure with the same block diagonal W_x (idem for B).

Multiplicative units: LSTM, attention mechanisms and variants

Some networks use multiplicative units: for instance, the basic building block of Long Short-Term Memory (LSTM) or gated recurrent units (GRUs) involve the following activation

$$\phi(v_1, v_2) := \phi_1(v_1)\phi_2(v_2),$$

where $v_1, v_2 \in \mathbb{R}$, and ϕ_1, ϕ_2 are *bounded* activations (for example tanh). The resulting function is Lipschitz-continuous: let c_i, γ_i denote the bounds and Lipschitz constants for ϕ_i, $i = 1, 2$ respectively. Then, the map ϕ is γ-Lipschitz, where $\gamma := c_1 c_2 \gamma_1 \gamma_2$; the condition $\lambda_{pf}(|A|) < 1/\gamma$ guarantees that the corresponding IDL model is well-posed. Similarly, attention models use component-wise vector multiplication, involving a softmax operation, as in

$$\phi(v_1, v_2) = \phi_1(v_1) \odot \mathrm{SoftMax}(v_2),$$

where ϕ_1 is a bounded activation. The activation ϕ has a Lipshitz constant w.r.t. the l_1-norm given by $\gamma = c_1\gamma_1 + 1$, with c_1 the bound on ϕ_1 and γ_1 its Lipschitz constant. Again, we can use the condition $\lambda_{pf}(|A|) < 1/\gamma$ to guarantee that the IDL model is well-posed.

7.5 Training Implicit Models

In a training context, we are given an input data matrix $U = [u_1, \ldots, u_m] \in \mathbb{R}^{p\times m}$ and response matrix $Y = [y_1, \ldots, y_m] \in \mathbb{R}^{q\times m}$, and seek to fit an IDL model, with the well-posedness condition $A \in \mathrm{WP}(\phi)$. Exploiting the result given in Theorem 7.3.2, we replace the well-posedness constraint by $\|A\|_\infty \leq r$, where $r \in [0, 1)$ is given. We write the training problem compactly as

$$\begin{aligned} \min_{A,B,C,D} \ & \mathcal{L}(Y, CX + DU) + \mathcal{P}(A, B, C, D) \\ \text{s.t. } & X = \phi(AX + BU), \ \ \|A\|_\infty \leq r. \end{aligned}$$

Here, $X = [x_1, \ldots, x_m] \in \mathbb{R}^{n \times m}$ contains all the hidden feature vectors for every data point; $\mathcal{L}$ is a summable loss function, assumed to be convex and differentiable in its second argument, and $\mathcal{P}$ is a convex penalty function, which can be used to enforce a given structure (such as, A strictly upper block triangular, or Toeplitz) on the parameters, and/or encourage their sparsity (using for instance l_1-norm penalties). We give some examples of loss functions and penalty functions in SM, §7.7.

We show that the objective is differentiable w.r.t. the parameters of the IDL model. By chain rule, this will be the case if the prediction $\hat{y}$ is differentiable w.r.t. (A, B, C, D). From the IDL model, we directly have that $\hat{y}$ is $\mathcal{C}^\infty$ w.r.t. (C, D) (linear function). Similarly to theorem 7.3.4 we have,

Theorem 7.5.1. *Consider an IDL model that satisfies the sufficient well-posedness condition $\lambda_{pf}(|A|) < 1$, then the prediction rule $A, B \to \hat{y}(A, B)$ is $\mathcal{C}^d$ if the activation ϕ is $\mathcal{C}^d$. The result also holds if the continuity properties of the activation are satisfied a.e.*

A proof is given in SM, §7.7. With that in mind, we can solve (7.5) using (stochastic) projected gradient descent. We provide a direct way of computing these gradients in SM, §7.7. As an example, the gradient w.r.t. matrix A with one data point (u, y), is given by

$$\nabla_A \mathcal{L}(y, \hat{y}) = \left(C(I - \tilde{D}A)^{-1}\tilde{D}\right)^\top \nabla_{\hat{y}} \mathcal{L}(y, \hat{y})\ x^\top,$$

where $\tilde{D} = \mathrm{diag}(\nabla\phi(Ax + Bu))$. Note that for (A, B, C, D) with the same structure as in Section 7.4 we recover the same gradient as backprop.

Due to the norm constraint, the gradient method requires a projection at each step. This step corresponds to a sub-problem of the form

$$\min_A \ \|A - A^0\|_F \ : \ \|A\|_\infty \leq r, \tag{7.4}$$

with $n \times n$ matrix A^0 given. The above problem is decomposable across rows (that is, features), and can be very efficiently solved using (vectorized) bisection, as detailed in SM, §7.7. All in all, the method we propose to learn the parameters of an IDL model is a stochastic projected gradient descent method (SPGD).

7.6 Numerical experiments

In this section, we provide preliminary results regarding the performance of IDL models using SPGD as presented in Section 7.5. We compare IDL models with FFNNs and show that the former has the potential of achieving better performance. To simplify the experiments, we apply no regularization to both IDL models and FFNNs.

Learning nonlinear functions via regression

We aim at learning a real function f with values $u \to f(u) = 5\cos(\pi u)\exp(-\frac{1}{2}|u|)$. We take the inputs at random with $m = 200$ and a hidden state of size $n = 75$. We consider the well-posedness condition $\|A\|_\infty \leq 1/2$. We update by block (A, B) and then (C, D). As shown in Figures 7.2 and 7.3, we are able to learn this function using our method. We compare to a FFNN trained using ADAM with three layers and similar hidden state size. We find that the IDL model slightly outperforms the FFNN on this task. More experimental design details and results are provided in SM, §7.7.

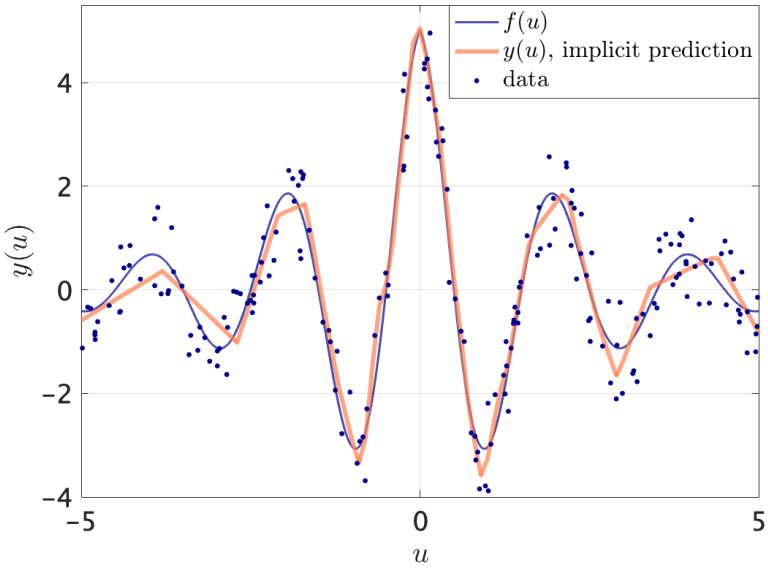

Figure 7.2: Implicit prediction $y(u)$ comparison with $f(u)$

MNIST dataset

For experiments on the MNIST dataset, we use an IDL model with $n = 100$ trained using SPGD ($\|A\|_\infty \leq 0.95$) and compare it to a three-layer FFNN (784-60-40-10) which has a similar hidden state size. In both cases, we use ReLU activations and softmax for the output. We train both models using cross entropy loss. The results for this experiment are given in Figure 7.4. We show that the IDL model slightly outperforms the FFNN.

German Traffic Sign Recognition dataset

The traffic sign classification benchmark (GTSRB) consists in 32×32 input datapoints, each input is associated to 43 possible traffic sign classes. We use an IDL model with $n = 400$

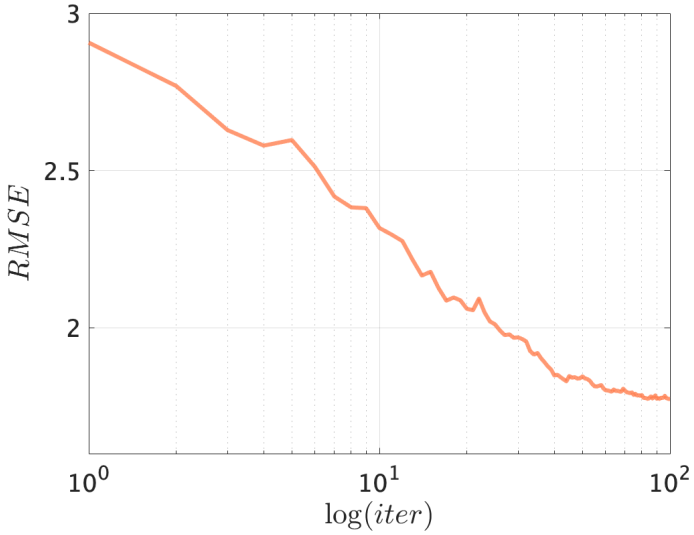

Figure 7.3: RMSE across stochastic projected gradient iterations for the (A, B) block updates

trained using SPGD ($\|A\|_\infty \leq 0.95$) and compare it to a three-layer FFNN (1024-300-100-43). In both cases, we use ReLU activations and softmax for the output. We train both models using cross entropy loss. The results for this experiment are given in Figure 7.5. Again, IDL model slightly outperforms the FFNN.

Conclusion

In this chapter we present a new type of deep learning model that is more general and includes popular deep learning models as special cases. Implicit models are notationally simple. They allow rigorous approaches to challenging problems in deep learning, ranging from robustness analysis, sparsity and interpretability, and feature selection. Here, we focus on the important well-posedness and composition issues, and derive a stochastic gradient algorithm for training.

Implicit models rely on a representation of the prediction rule where the linear operations are clearly separated from the (parameter-free) nonlinear ones, leading to a much simplified notation, as well as a higher capacity. Composition rules permit architectures that are not allowed under current deep learning paradigm, as the latter does not allow for cycles in the network. In particular, it is now possible to consider dynamical systems with neural networks as feedback controllers.

As a final note, consider Figure 7.6, which illustrates the striking similarity with two important models in systems and control theory: linear time-invariant systems [9] correspond to replacing activation map ϕ in IDL with an integration operator, while models for uncertain

Figure 7.4: Performance comparison on MNIST. Average best accuracy, implicit: 0.976, neural network: 0.972. The curves are generated from 5 different runs with the lines marked as mean and region marked as the standard deviation over the runs. Experimental details can be found in SM, §7.7

systems in robust control [46] involve (uncertain) norm bounded operators. Exploiting these connections is left for future work.

7.7 Proofs and additional notes

Additional notes on well-posedness

Proof of theorem 7.3.1

Let $b \in \mathbb{R}^n$. We first prove the existence of a solution $\xi \in \mathbb{R}^n$ to the equation $\xi = \phi(A\xi + b)$. Consider the Picard iteration (7.2). We have for every $t \geq 1$:

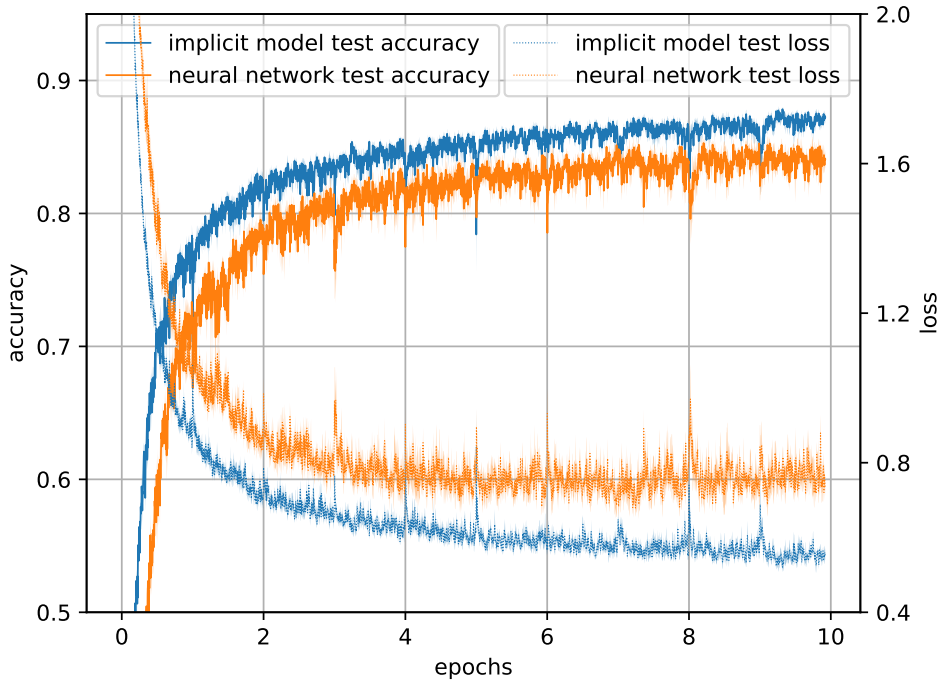

Figure 7.5: Performance comparison on GTSRB. Average best accuracy, implicit: 0.874, neural network: 0.859. The curves are generated from 5 different runs with the lines marked as mean and region marked as the standard deviation over the runs. Experimental details can be found in SM, §7.7.

$$\begin{aligned} |x(t+1) - x(t)| &= |\phi(Ax(t) + b) - \phi(Ax(t-1) + b)| \\ &\leq |A||x(t) - x(t-1)|, \end{aligned}$$

which implies that for every $t, h \geq 0$:

$$\begin{aligned} |x(t+\tau) - x(t)| &\leq \sum_{k=t}^{t+\tau} |A|^k |x(1) - x(0)| \\ &\leq |A|^t \sum_{k=0}^{\tau} |A|^k |x(1) - x(0)| \\ &\leq |A|^t w, \end{aligned}$$

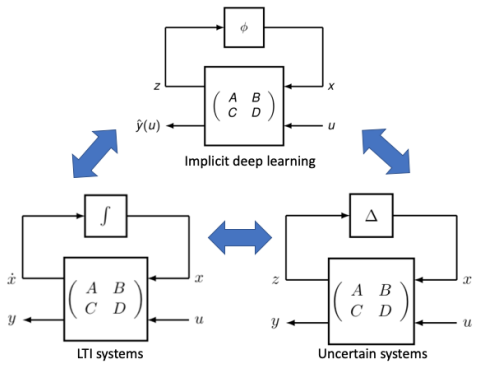

Figure 7.6: Some cousins of implicit models: LTI systems (bottom left) and uncertain systems (bottom right).

where

$$w := \sum_{k=0}^{+\infty} |A|^k |x(1) - x(0)| = (I - |A|)^{-1} |x(1) - x(0)|.$$

Here we exploited the fact that, due to $\lambda_{pf}(|A|) < 1$, $I - |A|$ is invertible, and the series above converges. Since $\lim_{t\to 0} |A|^t = 0$, we obtain that $x(t)$ is a Cauchy sequence, hence it has a limit point, x_∞. By continuity of ϕ we further obtain that $x_\infty = \phi(Ax_\infty + b)$, which establishes the existence of a solution.

To prove unicity, consider $x^1, x^2 \in \mathbb{R}^n_+$ two solutions to the equation. Using the hypotheses in the theorem, we have, for any $k \geq 1$:

$$|x^1 - x^2| \leq |A||x^1 - x^2| \leq |A|^k |x^1 - x^2|.$$

The fact that $|A|^k \to 0$ as $k \to +\infty$ then establishes unicity.

Proof of theorem 7.3.3

Express the equation $x = \phi(Ax + b)$ as

$$x_1 = \phi(A_{11}x_1 + A_{12}x_2 + b_1), \quad x_2 = \phi(A_{22}x_2 + b_2),$$

where $b = (b_1, b_2)$, $x = (x_1, x_2)$, with $b_i \in \mathbb{R}^{n_i}$, $x_i \in \mathbb{R}^{n_i}$, $i = 1, 2$. Here, since ϕ acts componentwise, we use the same notation ϕ in the two equations.

Now assume that A_{11} and A_{22} are well-posed w.r.t. ϕ. Since A_{22} is well-posed for ϕ, the second equation has a unique solution x_2^*; plugging $x_2 = x_2^*$ into the second equation, and using the well-posedness of A_{11}, we see that the first equation has a unique solution in x_1, hence A is well-posed.

To prove the converse direction, assume that A is well-posed. The second equation above must have a unique solution x_2^*, irrespective to the choice of b_2, hence A_{22} must be well-posed. To prove that A_{11} must be well-posed too, set $b_2 = 0$, b_1 arbitrary, leading to the system

$$x_1 = \phi(A_{11}x_1 + A_{12}x_2 + b_1), \quad x_2 = \phi(A_{22}x_2).$$

Since A_{22} is well-posed for ϕ, there is a unique solution x_2^* to the second equation; the first equation then reads $x_1 = \phi(A_{11}x_1 + b_1 + A_{12}x_2^*)$. It must have a unique solution for any b_1, hence A_{11} is well-posed.

Composition of implicit models

Implicit models can be easily composed via matrix algebra. Sometimes, the connection preserves well-posedness. We now discuss the cases when there are multiplicative connections and feedback connections.

Multiplicative connections are in general are not Lipschitz-continuous, unless the inputs are bounded. Precisely, consider two activation maps ϕ_i that are Lipschitz-continuous with constant γ_i and are bounded, with $|\phi_i(v)| \leq c_i$ for every v, $i = 1, 2$; then, the multiplicative map

$$(u_1, u_2) \in \mathbb{R}^2 \to \hat{y}(u) = \phi_1(u_1)\phi_2(u_2)$$

is Lipschitz-continuous w.r.t. the l_1-norm, with constant $\gamma := c_1\gamma_1 + c_2\gamma_2$.

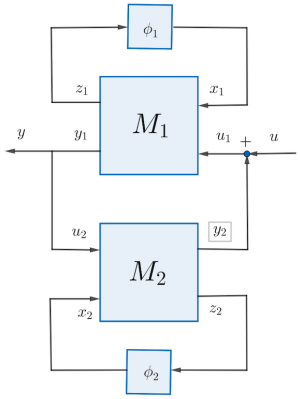

Figure 7.7: Feedback connection of two implicit models.

Finally, feedback connections are also possible. Consider two well-posed implicit systems:

$$y_i = C_i x_i + D_i u_i, \quad x_i = \phi_i(A_i x_i + B u_i), \quad i = 1, 2.$$

Now let us connect them in a feedback connection: the combined system is described by the IDL model, where $u_1 = u + y_2$, $u_2 = y_1 = y$.

Then, the feedback system is also an IDL model, with appropriate matrices (A, B, C, D), and activation map acting block-wise: $\phi(z_1, z_2) = (\phi_1(z_1), \phi_2(z_2))$ and state (x_1, x_2). In the simplified case when $D_1 = D_2 = 0$, the feedback connection has the model matrix

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cc|c} A_1 & B_1C_2 & B_1 \\ B_2C_1 & A_2 & 0 \\ \hline C_1 & 0 & 0 \end{array}\right).$$

Note that the connection is not necessarily well-posed.

Continuity and differentiability of implicit deep learning

Proof of Theorem 7.3.4

Let us define the function $F(u, x) = x - \phi(Ax + Bu)$. We immediately have F is continuous and that if the activation ϕ is $\mathcal{C}^d$ then F is also $\mathcal{C}^d$. This is the case for properties true a.e. Given $\tilde{u} \in \mathbb{R}^p$, let $\tilde{x}$ be the unique solution to the equilibrium equation (as $A \in \mathrm{WP}(\phi)$). We have $F(\tilde{u}, \tilde{x}) = 0$. Let us assume that F is differentiable w.r.t. x at $\tilde{x}$ (this holds a.e.), we have the Jacobian matrix

$$\nabla_x F(\tilde{u}, \tilde{x}) = I - A^\top \mathrm{diag}(\nabla\phi(A\tilde{x} + B\tilde{u}))$$

Using the fact that by non-expansiveness $|\nabla\phi(A\tilde{x} + B\tilde{u})| \leq 1$ and $\lambda_{pf}(|A|) < 1$ we have that $\nabla_x F(\tilde{u}, \tilde{x})$ is non-singular at $(\tilde{u}, \tilde{x})$. Then, using the implicit function theorem [19] (A.25), there exists open sets $S_{\tilde{x}} \subset \mathbb{R}^n$ and $S_{\tilde{u}} \subset \mathbb{R}^p$ containing $\tilde{x}$ and $\tilde{u}$, respectively, and a continuous function $\psi : S_{\tilde{u}} \to S_{\tilde{x}}$ such that $x = \psi(u)$ and $F(u, x) = 0 \iff x = \phi(Ax + Bu)$ for all $x \in S_{\tilde{x}}$ and $u \in S_{\tilde{u}}$. Furthermore, if ϕ is $\mathcal{C}^d$ then ψ is $\mathcal{C}^d$. As our analysis is local, we have that $u \to x(u)$ is $\mathcal{C}^d$ a.e. if ψ is $\mathcal{C}^d$ a.e. We finish the proof by simply observing that $\hat{y}(u) = Cx(u) + Du$, hence $\hat{y}$ inherits the continuity properties of $x(u)$.

Proof of Theorem 7.5.1

The proof for this Theorem is similar to the proof of 7.3.4, let us define $F(A, x) = x - \phi(Ax + c)$, with $c = Bu$ a constant, it then suffices to replace u by A in the previous proof. From the implicit function theorem we can also get a gradient of x as a function of A. We have $\nabla_A F(A, x) = -\nabla\phi(Ax + Bu)x^T$. Hence,

$$\nabla_A x(A) = \nabla\phi(Ax + c)x^\top \Big(I - A^\top \mathrm{diag}(\nabla\phi(Ax + c))\Big)^{-1}$$

which allows to get the same result as the one derived in SM §7.7.

Training Problem of Implicit Model

Examples of loss functions For regression tasks, we may use the squared Euclidean loss: for $Y, \hat{Y} \in \mathbb{R}^{q\times m}$,

$$\mathcal{L}(Y, \hat{Y}) := \frac{1}{2}\|Y - \hat{Y}\|_F^2.$$

For multi-class classification, a popular loss is a combination of negative cross-entropy with the soft-max: for two q-vectors $y, \hat{y}$, with $y \geq 0$, $y^\top 1 = 1$, we define

$$\mathcal{L}(y, \hat{y}) = -y^\top \log\left(\frac{e^{\hat{y}}}{\sum_{i=1}^q e^{\hat{y}_i}}\right) = \log(\sum_{i=1}^q e^{\hat{y}_i}) - y^\top \hat{y}.$$

We can extend the definition to matrices, by summing the contribution to all columns, each corresponding to a data point: for $Y, Z \in \mathbb{R}^{q\times m}$,

$$\begin{aligned}\mathcal{L}(Y, \hat{Y}) &= \sum_{j=1}^m \log\left(\sum_{i=1}^q e^{\hat{Y}_{ij}}\right) - \sum_{j=1}^m\sum_{i=1}^q Y_{ij}\hat{Y}_{ij} \\ &= \log(1^\top \exp(\hat{Y}))1 - \mathbf{Tr}Y^\top \hat{Y},\end{aligned}$$

where both the log and the exponential functions apply component-wise.

Examples of penalty functions Via an appropriate definition of $\mathcal{P}$, we can make sure that the model matrix A is well-posed w.r.t. ϕ, either imposing an upper triangular structure for A, or via n l_∞-norm constraint $\|A\|_\infty < 1$, or a Perron-Frobenius eigenvalue constraint. Note that in the case of the CONE maps such as the ReLU, due to scale invariance seen in theorem 7.3.2, we can always replace a Perron-Frobenius eigenvalue constraint with a l_∞-norm constraint.

Beyond well-posedness, the penalty can be used to encourage desired properties of the model. For robustness, the convex penalty (7.6) can be used, provided we also enforce $\|A\|_\infty < 1$. It follows from the fact that

$$\forall\, u, u^0 \;:\; \|\hat{y}(u) - \hat{y}(u^0)\|_\infty \leq \kappa\|u - u^0\|_\infty,$$

where

$$\kappa := \frac{\|B\|_\infty \cdot \|C\|_\infty}{1 - \|A\|_\infty} + \|D\|_\infty. \tag{7.5}$$

And thus we can arrive at a bound for (7.5).

$$\kappa \leq P(A, B, C, D) := \frac{1}{2}\frac{\|B\|_\infty^2 + \|C\|_\infty^2}{1 - \|A\|_\infty} + \|D\|_\infty. \tag{7.6}$$

Gradient Descent Equations

In this section, we explain how one can compute gradients w.r.t. M through an equilibrium equation. In the sequel, we consider mini-batches of size 1 (that is, $X = x \in \mathbb{R}^n$ and $U = u \in \mathbb{R}^p$), in order to simplify the notation. For convenience, we use $\hat{y} = Cx + Du$, $z = Ax + Bu$. We also assume that the map ϕ is differentiable.

We wish to calculate

$$\nabla_M \mathcal{L} = \begin{pmatrix} \nabla_A \mathcal{L} & \nabla_B \mathcal{L} \\ \nabla_C \mathcal{L} & \nabla_D \mathcal{L} \end{pmatrix}$$

The difficult part of the above calculation is $\nabla_A \mathcal{L}$ and $\nabla_B \mathcal{L}$ due to the presence of the implicit equality constraint. We deal with this via implicit differentiation.

Calculating $\nabla_A \mathcal{L}$ We have that

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial A_{jk}} &= \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial Ax + Bu}{\partial A_{jk}} \\ &= \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial \sum_m A_{lm} x_m}{\partial A_{jk}} \\ &= \frac{\partial \mathcal{L}}{\partial z} e_j x_k \\ &= \left[\nabla_z \mathcal{L}\, x^\top\right]_{jk} \\ \nabla_A \mathcal{L} &= \nabla_z \mathcal{L}\, x^\top \end{aligned}$$

We calculate $\nabla_z \mathcal{L}$ via implicit differentiation:

$$\begin{aligned} \nabla_z \mathcal{L} &= \left(\frac{\partial \mathcal{L}}{\partial x} \cdot \frac{\partial x}{\partial z}\right)^\top, \\ \frac{\partial \mathcal{L}}{\partial x} &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial Cx + Du}{\partial x}, \\ \frac{\partial x}{\partial z} &= \frac{\partial \phi(z)}{\partial z} + \frac{\partial \phi(Ax + Bu)}{\partial x} \cdot \frac{\partial x}{\partial z} \\ \frac{\partial x}{\partial z} &= (I - \tilde{D}A)^{-1} \tilde{D}, \end{aligned} \tag{7.7}$$

where $\tilde{D} = \nabla_z \phi(z)$ is a diagonal matrix. A question that naturally arises is the existence of an inverse for matrix $(I - \tilde{D}A)$. For component-wise non expansive maps ϕ, we have that $I \succ \tilde{D}$, and considering th PF well-posedness condition we directly get $\lambda_{pf}(|\tilde{D}A|) \leq 1$. We also have $\|\tilde{D}A\|_\infty \leq \|A\|_\infty$ and $\|\tilde{D}A\|_1 \leq \|A\|_1$, therefore for other well-posedness condition this inverse also exists. Note that $\frac{\partial \mathcal{L}}{\partial \hat{y}}$ (and hence $\nabla_{\hat{y}} \mathcal{L}$) is the gradient of the loss function and thus can be easily computed.

Calculating $\nabla_B \mathcal{L}$ From above it follows that

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial B_{jk}} &= \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial Ax + Bu}{\partial B_{jk}} \\ &= \left[\nabla_z \mathcal{L}\, u^\top\right]_{jk} \\ \nabla_B \mathcal{L} &= \nabla_z \mathcal{L}\, u^\top,\end{aligned}$$

where $\nabla_z \mathcal{L}$ is given in equation (7.7).

Calculating $\nabla_C \mathcal{L}$ Similar to above, it follows that

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial C_{jk}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial Cx + Du}{\partial C_{jk}} \\ &= \left[\nabla_{\hat{y}} \mathcal{L}\, x^\top\right]_{jk} \\ \nabla_C \mathcal{L} &= \nabla_{\hat{y}} \mathcal{L}\, x^\top.\end{aligned}$$

Calculating $\nabla_D \mathcal{L}$ Similar to above, it follows that

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial D_{jk}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial Cx + Du}{\partial D_{jk}} \\ &= \left[\nabla_{\hat{y}} \mathcal{L}\, u^\top\right]_{jk} \\ \nabla_D \mathcal{L} &= \nabla_{\hat{y}} \mathcal{L}\, u^\top.\end{aligned}$$

Calculating $\nabla_u \mathcal{L}$ Additionally, implicit differentiation enables differentiation through the implicit model by obtaining the gradient w.r.t. input u.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial u_j} &= \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial Ax + Bu}{\partial u_j} + \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial Cx + Du}{\partial u_j} \\ &= \frac{\partial \mathcal{L}}{\partial z} \cdot Be_j + \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot De_j \\ &= \left[B^\top \nabla_z \mathcal{L} + D^\top \nabla_{\hat{y}} \mathcal{L}\right]_j \\ \nabla_u \mathcal{L} &= B^\top \nabla_z \mathcal{L} + D^\top \nabla_{\hat{y}} \mathcal{L},\end{aligned}$$

where $\nabla_z \mathcal{L}$ is given in equation (7.7).

Projection on l_∞ Matrix Norm Ball

We address problem (7.4), which we write as

$$p^* := \min_A \ \frac{1}{2}\|A - A^0\|_F^2 \ : \ \sum_{j \in [n]} |A_{ij}| \leq \kappa, \ \ i \in [n].$$

where $A^0 \in \mathbb{R}^{n\times n}$ is given. The problem is decomposable across the rows of the matrices involved, leading to n sub-problems of the form

$$\min_a \ \frac{1}{2}\|a - a_i^0\|_2^2 \ : \ \|a\|_1 \leq \kappa,$$

which $a_i^0 \in \mathbb{R}^n$ the i-th row of A^0.

The problem cannot be solved in closed form, but a bisection method can be applied to the dual:

$$p^* = \max_{\lambda\geq 0} \ -\kappa\lambda + \sum_{i\in[n]} s_i(\lambda),$$

where, for $\lambda \geq 0$ given:

$$s_i(\lambda) := \min_\xi \ \frac{1}{2}(\xi - a_i^0)^2 + \lambda|\xi|, \ \ i \in [n].$$

A subgradient of the objective is

$$g_i(\lambda) := -\kappa + \sum_{i\in[n]} \max(|a_i^0| - \lambda, 0), \ \ i \in [n].$$

Observe that $p^* \geq 0$, hence at optimum:

$$0 \leq \lambda \leq \frac{1}{\kappa}\sum_{i\in[n]} s(\lambda, a_i^0) \leq \lambda^{\max} := \frac{1}{2\kappa}\|a_i^0\|_2^2.$$

The bisection can be initialized with the interval $\lambda \in [0, \lambda^{\max}]$.

Returning to the original problem (7.4), we see that all the iterations can be expressed in a "vectorized" form, where updates for the different rows of A are done in parallel. The dual variables corresponding to each row are collected in a vector $\lambda \in \mathbb{R}^n$. We initialize the bisection with a vector interval $[\lambda_l, \lambda_u]$, with $\lambda^l = 0$, $\lambda_i^u = \frac{1}{2}\|a_i^0\|_2^2/\kappa$, $i \in [n]$. We update the current vector interval as follows:

1. Set $\lambda = (\lambda_l + \lambda_u)/2$.
2. Form a vector $g(\lambda)$ containing the sub-gradients corresponding to each row, evaluated at λ_i, $i \in [n]$:
$$g(\lambda) = -\kappa 1 + (|A^0| - \lambda 1^\top)_+^\top 1.$$
3. For every $i \in [n]$, reset $\lambda_i^u = \lambda_i$ if $g_i(\lambda) > 0$, $\lambda_i^l = \lambda_i$ if $g_i(\lambda) \leq 0$.

Numerical experiments

Learning nonlinear function via regression

We take $m = 200$ and draw the inputs u_i, $i \in [m]$ uniformly at random between -5 and 5. We add random noise to the output, $y(u) = f(u) + w$ with w taken uniformly at random between -1 and 1, hence the standard deviation for $y(u)$ is $1/\sqrt{3} \simeq 0.57$. As a loss function we use the squared euclidean loss presented in Section 7.7. We take $n = 75$ and we learn the implicit model by doing only two block updates: first, we update (A, B) using stochastic projected gradient descent. The RMSE across iterations is shown in Figure **??**. After this first update we achieve a RMSE of 1.77. We then update (C, D), this step only consists in a linear regression problem. After this update we achieve a RMSE of 0.56. For comparison we also train a neural network with 3 hidden layers of width $n/3 = 25$ using ADAM (run until convergence), with mini-batches and a tuned learning rate. We run ADAM until convergence. We get an RMSE= 0.65 that is slightly above that of the implicit model.

MNIST

The MNIST dataset consists in input data points that are 28×28 images (8bit gray scale for each pixel) of hand written digits from 0 to 9. Each image is reshaped into a 784-dimensional vector and rescaled to values between 0 and 1 before training. There are 50000 training data points and 10000 testing data points. The goal is to classify correctly the digit of the labelled input image. For training, we employ SPGD for the IDL model and ADAM with learning rate of 5e-3 for both models (tuned through grid search from 1e-1 to 1e-4 that offer best test performance) and trained over cross entropy loss. And the performance is examined through the accuracy and loss on the test data set over the training process. In both cases we use a batchsize of 100.

GTSRB dataset

In the GTSRB dataset, the input data points are 32×32 images with rgb channels of traffic signs corresponding to 43 classes. Each image is turned into gray scale before being reshaped into a 1024 dimensional vector and then re-scaled to be between 0 and 1. There are 34,799 training data points and 12,630 testing data points. The goal is to correctly classify the traffic signs. For training, we employ SPGD for the IDL model and ADAM with learning rate of 5e-3 for FFNN and 1e-3 for the IDL model (tuned through grid search from 1e-1 to 1e-4 that offer best test performance) and trained over cross entropy loss. And the performance is examined through the accuracy and loss on the test data set over the training process.

Chapter 8

Brief directions for future works

This thesis was organized into three parts. In the first part we focused on optimization models and method for participation of DERs in electricity markets; in the second part we studied Hopfield methods as a heuristic to solve combinatorial problems; and in the third part we developed a general framework for implicit optimization and implicit deep learning.

In part I, all the models we used were convex. We already mentioned that some control variables are in fact discrete variables (and therefore non-convex) when we introduced Hopfield methods. More generally speaking, there are many variables in energy applications that should be considered as discrete variables: for instance in the smart home realm, the control of PEV charge, pool pumps and refrigerators. Note that we also studied in part II an economic load dispatch problem where one of the decision variable corresponded to turning on or off a power plant. An area for future work, would be to use the Hopfield methods we presented (or the Fenchel ADMM approach of part III) to coordinate DERs with discrete states.

Discrete variables are a source of non-convexity, but there are many other ways to improve the model used for DERs: among others, nonlinearities. For instance, a nonlinear model for the charge of Li-ion batteries could be considered. For such nonlinearities, it would be of interest to use the universal approximation approach (i.e. model the nonlinearity as a feedforward neural network with one layer) and the Fenchel divergence penalty. Remark that we already did something similar in the example we had on PEV parking overstay in part III. We believe the methods we presented could more generally be used for energy systems models with nonlinearities (in the dynamic or the objective) to create algorithms that perform better than some of the methods in use currently (e.g. sequential quadratic programming).

Similarly, in the first chapter we considered a Markowitz portfolio objective, which is quadratic and convex. Remark, that if the aggregator participation in the electricity market is high, the DER coordinator will have an impact on the prices themselves. Therefore the DER coordinator could take this phenomenon into account by modeling price prediction not only as a function of time (or weather conditions) but also as a function of its own supply/demand in the market - which would make the objective non-convex in most cases. Research in that area would involve finding a way to represent and learn this mapping (maybe an implicit model?) as well as variations to the algorithms we proposed (maybe Fenchel ADMM?).

Without considering prices to be endogenous (i.e. dependent on the aggregator market participation), we believe another area of research would consist in improving the risk model for the market prices. For instance, the practical performance of different risk measures (potentially non-convex measures) than that of Markowitz portfolios could be examined. Remark that changing the risk measure would in turn change the behavior of the algorithms - such as the convergence rates - we presented. Eventually new algorithms for distributing the computation burden to prosumers would be required.

In part II, we developed algorithms and theory for combinatorial optimization problems. We believe that future work could consist in applying these methods to DER integration problems as mentioned above, as well as more general problems such as Hybrid systems control. We showed some promising first results for these methods, but their behavior and performance on more practical problems and datasets still need to be assessed, particularly given the limitations we pointed out at the end of part II. Some of these limitations could also be lifted: for instance we mentioned that Hopfield methods could be sensitive to initial conditions. Hence, finding a good starting point method given the geometry and behavior of Hopfield methods could further improve the quality of the candidate solution. We also mentioned that another limitation we faced was the dual algorithm (i.e. the dual variable update) that we use to handle inequality constraints and the choice of step size. We do not believe we have found a robust way to guarantee convergence or guarantee in practice the quality of the candidate solution. For the moment, this method needs to be careful tuning given an application or a combinatorial optimization problem. Therefore, crafting better (e.g. requiring less hyperparameter tuning) dual algorithms for Hopfield methods could be a valuable research endeavour.

Similarly to Hopfield methods, we believe that future research in implicit optimization could consist at first in applying these implicit methods on more practical problems and datasets. We also understand that the applications we found (combinatorial optimization, nonlinear control and deep learning) are probably not the only ones, and that the algorithms we developed are just one way for solving such problems. All of this offers a large space for new discoveries using implicit optimization. Moreover, in the last chapter we saw that implicit optimization included implicit deep learning as a special case, and that most deep learning architectures were themselves special cases of implicit architectures. From that, we hope to convey the richness the implicit optimization framework has to offer, and hope that even more research avenues in that area will be explored.

Bibliography

[1] 2015. URL: https://jancovici.com/en/energy-transition/energy-and-us/what-is-energy-actually/.

[2] Shakeb Afsah, Kendyl Salcito, and Chris Wielga. "Energy efficiency is for real, energy rebound a distraction". In: *CO2 Scorecard, Research Notes* (2012).

[3] SVB Aiyer and Frank Fallside. *A Subspace Approach to Solving Combinatorial Optimization Problems with Hopfield Networks*. University of Cambridge, Department of Engineering, 1990.

[4] Mohamed H Albadi and Ehab F El-Saadany. "A summary of demand response in electricity markets". In: *Electric power systems research* 78.11 (2008), pp. 1989–1996.

[5] Brandon Amos and J Zico Kolter. "Optnet: Differentiable optimization as a layer in neural networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 136–145.

[6] Brandon Amos et al. "Differentiable MPC for End-to-end Planning and Control". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8289–8300.

[7] Salla Annala et al. "Does knowledge contribute to the acceptance of demand response?" In: *Journal of Sustainable Development of Energy, Water and Environment Systems* 2.1 (2014), pp. 51–60.

[8] AnonymousAuthors. "Implicit Deep Learning". In preparation, to be submitted to SIMODS. 2020.

[9] J Dwight Aplevich. *The essentials of linear state-space systems*. Wiley New York, 2000.

[10] Martin Arjovsky et al. "Invariant risk minimization". In: *arXiv preprint arXiv:1907.02893* (2019).

[11] Armin Askari et al. "Lifted neural networks". In: *arXiv preprint arXiv:1805.01532* (2018).

[12] Lawrence M Ausubel and Peter Cramton. "Virtual power plant auctions". In: *Utilities Policy* 18.4 (2010), pp. 201–208.

[13] Filipe de Avila Belbute-Peres et al. "End-to-End Differentiable Physics for Learning and Control". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 7178–7189.

[14] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. "Deep Equilibrium Models". Preprint submitted. 2019.

[15] John Bates and David Leibling. "Spaced out". In: *Perspectives on parking policy* 9 (2012).

[16] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[17] Stephen R Becker, Emmanuel J Candès, and Michael C Grant. "Templates for convex cone problems with applications to sparse signal recovery". In: *Mathematical programming computation* 3.3 (2011), p. 165.

[18] Abraham Berman and Robert J Plemmons. *Nonnegative matrices in the mathematical sciences*. SIAM, 1994.

[19] Dimitri P Bertsekas. "Nonlinear programming". In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334.

[20] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 2016.

[21] Ricardo J Bessa et al. "Optimized bidding of a EV aggregation agent in the electricity market". In: *IEEE Transactions on Smart Grid* 3.1 (2011), pp. 443–452.

[22] Abhinav Bhattacharyya et al. "Nudging people towards more sustainable residential choice decisions: an intervention based on focalism and visualization". In: *Transportation* 46.2 (2019), pp. 373–393.

[23] Christian Bliek1ú, Pierre Bonami, and Andrea Lodi. "Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report". In: *Proceedings of the twenty-sixth RAMP symposium*. 2014, pp. 16–17.

[24] Alberto A Boretti. "Improvements of vehicle fuel economy using mechanical regenerative braking". In: *International journal of vehicle design* 55.1 (2011), pp. 35–48.

[25] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[27] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[28] Fiona Burlig et al. "Low energy: Estimating electric vehicle electricity use". In: *AEA Papers and Proceedings*. Vol. 111. 2021, pp. 430–35.

[29] *California Solar*. URL: `https://www.seia.org/state-solar-policy/california-solar`.

[30] Pierre Cardaliaguet and Guillaume Euvrard. "Approximation of a function and its derivative with a neural network". In: *Neural Networks* 5.2 (1992), pp. 207–220.

[31] Miguel Carreira-Perpinan and Weiran Wang. "Distributed optimization of deeply nested systems". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, 2014, pp. 10–19. URL: http://proceedings.mlr.press/v33/carreira-perpinan14.html.

[32] Andre Carrel, Raja Sengupta, and Joan L Walker. "The San Francisco Travel Quality Study: tracking trials and tribulations of a transit taker". In: *Transportation* 44.4 (2017), pp. 643–679.

[33] Bipartisan Policy Center. "Annual Energy Outlook 2020". In: *Energy Information Administration, Washington, DC* (2020).

[34] Shihua Chen, Qin Zhang, and Changping Wang. "Existence and stability of equilibria of the continuous-time Hopfield neural network". In: *Journal of Computational and Applied Mathematics* 169.1 (2004), pp. 117–125.

[35] Tian Qi Chen et al. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 6571–6583.

[36] Zhi Chen, Lei Wu, and Yong Fu. "Real-time price-based demand response management for residential appliances via stochastic optimization and robust optimization". In: *IEEE Transactions on Smart Grid* 3.4 (2012), pp. 1822–1831.

[37] Kristien Clement, Edwin Haesen, and Johan Driesen. "Coordinated charging of multiple plug-in hybrid electric vehicles in residential distribution grids". In: *2009 IEEE/PES Power Systems Conference and Exposition*. IEEE. 2009, pp. 1–7.

[38] LCG Consulting. *EnergyOnline, CAISO: Average Price and Demand*. 2017.

[39] Richard W Cottle, Franco Giannessi, and Jacques Louis Lions. *Variational inequalities and complementarity problems: theory and applications*. John Wiley & Sons, 1980.

[40] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[41] John Darzentas. "Problem complexity and method efficiency in optimization". In: *Journal of the Operational Research Society* 35.5 (1984), pp. 455–455.

[42] Paul Denholm et al. *Overgeneration from solar energy in california. a field guide to the duck chart*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2015.

[43] *Diablo Canyon Power Plant*. 2021. URL: https://en.wikipedia.org/wiki/Diablo_Canyon_Power_Plant.

[44] Priya Donti, Brandon Amos, and J Zico Kolter. "Task-based end-to-end model learning in stochastic optimization". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5484–5494.

[45] John Duchi et al. "Efficient projections onto the l 1-ball for learning in high dimensions". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 272–279.

[46] Geir E Dullerud and Fernando Paganini. *A course in robust control theory: a convex approach*. Vol. 36. Springer Science & Business Media, 2013.

[47] Laurent El Ghaoui et al. "Implicit deep learning". In: *arXiv preprint arXiv:1908.06315* (2019).

[48] Antony Evans and Andreas Schäfer. "The rebound effect in the aviation sector". In: *Energy economics* 36 (2013), pp. 158–165.

[49] *FACT SHEET: Biden Administration Advances Electric Vehicle Charging Infrastructure*. 2021. URL: https://www.whitehouse.gov/briefing-room/statements-releases/2021/04/22/fact-sheet-biden-administration-advances-electric-vehicle-charging-infrastructure/.

[50] *FACT SHEET: President Biden Sets 2030 Greenhouse Gas Pollution Reduction Target Aimed at Creating Good-Paying Union Jobs and Securing U.S. Leadership on Clean Energy Technologies*. 2021. URL: https://www.whitehouse.gov/briefing-room/statements-releases/2021/04/22/fact-sheet-president-biden-sets-2030-greenhouse-gas-pollution-reduction-target-aimed-at-creating-good-paying-union-jobs-and-securing-u-s-leadership-on-clean-energy-technologies/.

[51] Roger Fletcher and Sven Leyffer. "Numerical experience with lower bounds for MIQP branch-and-bound". In: *SIAM Journal on Optimization* 8.2 (1998), pp. 604–616.

[52] Marguerite Frank and Philip Wolfe. "An algorithm for quadratic programming". In: *Naval research logistics quarterly* 3.1-2 (1956), pp. 95–110.

[53] Steven A Gabriel et al. *Complementarity modeling in energy markets*. Vol. 180. Springer Science & Business Media, 2012.

[54] Bolin Gao and Lacra Pavel. "On the properties of the softmax function with application in game theory and reinforcement learning". In: *arXiv preprint arXiv:1704.00805* (2017).

[55] Andrew H Gee and Richard W Prager. "Polyhedral combinatorics and neural networks". In: *Neural Computation* 6.1 (1994), pp. 161–180.

[56] Christina Gerhardt. "Back to the Future or Forward to the Past: Ocean Voyaging and Slow Travel." In: *Studies in Twentieth and Twenty-First Century Literature* 44.1 (2020), COV1–COV1.

[57] Brian P Gerkey and Maja J Mataric. "A framework for studying multi-robot task allocation". In: (2003).

[58] Kenneth Gillingham et al. "The rebound effect is overplayed". In: *Nature* 493.7433 (2013), pp. 475–476.

[59] Fred Glover. "Future paths for integer programming and links to artificial intelligence". In: *Computers & operations research* 13.5 (1986), pp. 533–549.

[60] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[61] Tom Goldstein, Christoph Studer, and Richard Baraniuk. "A field guide to forward-backward splitting with a FASTA implementation". In: *arXiv preprint arXiv:1411.3406* (2014).

[62] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations*. 2015. URL: `http://arxiv.org/abs/1412.6572`.

[63] Andrzej Granas and James Dugundji. *Fixed point theory*. Springer Science & Business Media, 2013.

[64] Michael Grant, Stephen Boyd, and Yinyu Ye. *CVX: Matlab software for disciplined convex programming*. 2008.

[65] Julia B Griswold et al. "A behavioral modeling approach to bicycle level of service". In: *Transportation research part A: policy and practice* 116 (2018), pp. 166–177.

[66] Fangda Gu, Armin Askari, and Laurent El Ghaoui. "Fenchel lifted networks: A Lagrange relaxation of neural network training". In: *arXiv preprint arXiv:1811.08039* (2018).

[67] Zhi-Hong Guan, Guanrong Chen, and Yi Qin. "On equilibria, stability, and instability of Hopfield neural networks". In: *IEEE Transactions on Neural Networks* 11.2 (2000), pp. 534–540.

[68] Abhishek Halder et al. "Hopfield Neural Network Flow: A Geometric Viewpoint". In: *arXiv preprint arXiv:1908.01270* (2019).

[69] Abhishek Halder et al. "Hopfield Neural Network Flow: A Geometric Viewpoint". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[70] Rögnvaldur Hannesson. "Energy and GDP growth". In: *International Journal of Energy Sector Management* (2009).

[71] Shubhi Harbola and Volker Coors. "One dimensional convolutional neural network architectures for wind prediction". In: *Energy Conversion and Management* 195 (2019), pp. 70–75.

[72] Michael Z Hauschild, Ralph K Rosenbaum, and Stig Irving Olsen. *Life cycle assessment*. Vol. 2018. Springer, 2018.

[73] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[74] Leif Hockstad and L Hanel. *Inventory of US greenhouse gas emissions and sinks*. Tech. rep. Environmental System Science Data Infrastructure for a Virtual Ecosystem, 2018.

[75] JH Holland. "Adaptation in natural and artificial systems. MI". In: *Ann Arbor: University of Michigan Press* (1975).

[76] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[77] John J Hopfield. "Neurons with graded response have collective computational properties like those of two-state neurons". In: *Proceedings of the national academy of sciences* 81.10 (1984), pp. 3088–3092.

[78] John J Hopfield and David W Tank. ""Neural" computation of decisions in optimization problems". In: *Biological cybernetics* 52.3 (1985), pp. 141–152.

[79] Kejia Hu and Yuche Chen. "Technological growth of fuel efficiency in european automobile market 1975–2015". In: *Energy Policy* 98 (2016), pp. 142–148.

[80] Matthias Huber, Desislava Dimkova, and Thomas Hamacher. "Integration of wind and solar power in Europe: Assessment of flexibility requirements". In: *Energy* 69 (2014), pp. 236–246.

[81] Jeff St. John. *The California Duck Curve Is Real, and Bigger Than Expected*. 2016. URL: `https://www.greentechmedia.com/articles/read/the-california-duck-curve-is-real-and-bigger-than-expected`.

[82] B Kamgar-Parsi and B Kamgar-Parsi. "Clustering taxonomic data with neural networks". In: *Proceedings of the International Neural Network Conference—Washington DC*. Vol. 1. 1990, pp. 277–80.

[83] Kazuya Kawakami. "Supervised sequence labelling with recurrent neural networks". In: *Ph. D. thesis* (2008).

[84] Michaeel Kazi and Brian Thompson. "Implicitly-defined neural networks for sequence labeling". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2017, pp. 172–177.

[85] James Kennedy and Russell Eberhart. "PSO optimization". In: *Proc. IEEE Int. Conf. Neural Networks*. Vol. 4. IEEE Service Center, Piscataway, NJ. 1995, pp. 1941–1948.

[86] Byung-Gook Kim et al. "Dynamic pricing and energy consumption scheduling with reinforcement learning". In: *IEEE Transactions on Smart Grid* 7.5 (2015), pp. 2187–2198.

[87] David L King, James K Dudley, and William E Boyson. "PVSIM/sub C: a simulation program for photovoltaic cells, modules, and arrays". In: *Conference Record of the Twenty Fifth IEEE Photovoltaic Specialists Conference-1996*. IEEE. 1996, pp. 1295–1297.

[88] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[89] Uroš Klanšek. "Solving the nonlinear discrete transportation problem by MINLP optimization". In: *Transport* 29.1 (2014), pp. 1–11.

[90] J. Kolter. "Personal communication with A. Askari". Aug. 2019.

[91] Walid Krichene, Alexandre Bayen, and Peter L Bartlett. "Accelerated mirror descent in continuous and discrete time". In: *Advances in neural information processing systems*. 2015, pp. 2845–2853.

[92] Sergej B Kuksin and Sergej Kuksin. *Analysis of hamiltonian PDEs*. Vol. 19. Clarendon Press, 2000.

[93] Elizaveta Kuznetsova et al. "Reinforcement learning for microgrid energy management". In: *Energy* 59 (2013), pp. 133–146.

[94] Caroline Le Floch, Francois Belletti, and Scott Moura. "Optimal charging of electric vehicles for load shaping: A dual-splitting framework with explicit convergence bounds". In: *IEEE Transactions on Transportation Electrification* 2.2 (2016), pp. 190–199.

[95] Caroline Le Floch, Florent Di Meglio, and Scott Moura. "Optimal charging of vehicle-to-grid fleets via pde aggregation techniques". In: *2015 American Control Conference (ACC)*. IEEE. 2015, pp. 3285–3291.

[96] Jon Lee. *A first course in combinatorial optimization*. 36. Cambridge University Press, 2004.

[97] R Lempert et al. "Pathways to 2050: Alternative Scenarios for Decarbonizing the US Economy". In: *Center for Climate and Energy Solutions (C2ES), Arlington, Virginia. Available from: https://www. c2es. org/site/assets/uploads/2019/05/pathways-to-2050-scenariosfor-decarbonizing-the-us-economy-final. pdf.(Accessed 25 June 2019)* (2019).

[98] Jia Li, Cong Fang, and Zhouchen Lin. "Lifted proximal operator machines". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 4181–4188.

[99] Ling-Ling Li et al. "Renewable energy prediction: A novel short-term prediction model of photovoltaic output power". In: *Journal of Cleaner Production* 228 (2019), pp. 359–375.

[100] Thomas Lipp and Stephen Boyd. "Variations and extension of the convex–concave procedure". In: *Optimization and Engineering* 17.2 (2016), pp. 263–287.

[101] Chee-Kit Looi. "Neural network methods in combinatorial optimization". In: *Computers & Operations Research* 19.3-4 (1992), pp. 191–208.

[102] László Lovász. "On the ratio of optimal integral and fractional covers". In: *Discrete mathematics* 13.4 (1975), pp. 383–390.

[103] László Lovász. "On the Shannon capacity of a graph". In: *IEEE Transactions on Information theory* 25.1 (1979), pp. 1–7.

[104] Zhou Lu et al. "The expressive power of neural networks: A view from the width". In: *arXiv preprint arXiv:1709.02540* (2017).

[105] Juan M Lujano-Rojas et al. "Optimum residential load management strategy for real time pricing (RTP) demand response programs". In: *Energy policy* 45 (2012), pp. 671–679.

[106] *Maps and Data - Annual Vehicle Miles Traveled in the United States*. URL: `https://afdc.energy.gov/data/10315`.

[107] Johan Mathe et al. "PVNet: A LRCN architecture for spatio-temporal photovoltaic PowerForecasting from numerical weather prediction". In: *arXiv preprint arXiv:1902.01453* (2019).

[108] Rahul Mazumder and Trevor Hastie. "The graphical lasso: New insights and alternatives". In: *Electronic journal of statistics* 6 (2012), p. 2125.

[109] Garth P McCormick. "The projective SUMT method for convex programming". In: *Mathematics of operations research* 14.2 (1989), pp. 203–223.

[110] MM Monteiro, JE Leal, and FMP Raupp. "A four-type decision-variable MINLP model for a supply chain network design". In: *Mathematical Problems in Engineering* 2010 (2010).

[111] Scott J Moura, Nalin A Chaturvedi, and Miroslav Krstić. "Adaptive partial differential equation observer for battery state-of-charge/state-of-health estimation via an electrochemical model". In: *Journal of Dynamic Systems, Measurement, and Control* 136.1 (2014).

[112] Scott J Moura, Jeffrey L Stein, and Hosam K Fathy. "Battery-health conscious power management in plug-in hybrid electric vehicles via electrochemical modeling and stochastic control". In: *IEEE Transactions on Control Systems Technology* 21.3 (2012), pp. 679–694.

[113] Yu Nesterov. "A method of solving a convex programming problem with convergence rate O (1/kˆ 2) O (1/k2)". In: *Sov. Math. Dokl.* Vol. 27. 2.

[114] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate O (1/k2)". In: *Soviet Mathematics Doklady*. Vol. 27. 2. 1983, pp. 372–376.

[115] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.

[116] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[117] Daniel O'Neill et al. "Residential demand response using reinforcement learning". In: *2010 First IEEE international conference on smart grid communications*. IEEE. 2010, pp. 409–414.

[118] Brendan O'donoghue and Emmanuel Candes. "Adaptive restart for accelerated gradient schemes". In: *Foundations of computational mathematics* 15.3 (2015), pp. 715–732.

[119] Mattias Ohlsson, Carsten Peterson, and Bo Söderberg. "Neural networks for optimization problems with inequality constraints: the knapsack problem". In: *neural computation* 5.2 (1993), pp. 331–339.

[120] Obed Nelson Onsomu and Bülent Yeşilata. "Virtual power plant application for rooftop photovoltaic systems". In: *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE. 2019, pp. 1–5.

[121] Jaehyun Park and Stephen Boyd. "General heuristics for nonconvex quadratically constrained quadratic programming". In: *arXiv preprint arXiv:1703.07870* (2017).

[122] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[123] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *Ussr computational mathematics and mathematical physics* 4.5 (1964), pp. 1–17.

[124] Erik Porse et al. "Net solar generation potential from urban rooftops in Los Angeles". In: *Energy Policy* 142 (2020), p. 111461.

[125] Danny Pudjianto, Charlotte Ramsay, and Goran Strbac. "Virtual power plant and system integration of distributed energy resources". In: *IET Renewable power generation* 1.1 (2007), pp. 10–16.

[126] Ye QI, Jiaqi Lu, and Mengye Zhu. *Wind curtailment in China and lessons from the United States*. 2018. URL: `https://www.brookings.edu/research/wind-curtailment-in-china-and-lessons-from-the-united-states/`.

[127] Prabhakar Raghavan and Clark D Tompson. "Randomized rounding: a technique for provably good algorithms and algorithmic proofs". In: *Combinatorica* 7.4 (1987), pp. 365–374.

[128] J Ramanujam and P Sadayappan. "Optimization by neural networks". In: *IEEE International Conference on Neural Networks*. Vol. 2. 1988, pp. 325–332.

[129] Peter Richardson, Damian Flynn, and Andrew Keane. "Local versus centralized charging strategies for electric vehicles in low voltage distribution systems". In: *IEEE Transactions on Smart Grid* 3.2 (2012), pp. 1020–1028.

[130] Eckehard Rosenbaum. "Rebound effects and green growth–An examination of their relationship in a parsimonious equilibrium input-output-framework". In: *Journal of cleaner production* 225 (2019), pp. 121–132.

[131] Mark Ruth et al. *Effects of home energy management systems on distribution utilities and feeders under various market structures*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2015.

[132] Hedayat Saboori, M Mohammadi, and R Taghe. "Virtual power plant (VPP), definition, concept, components and types". In: *2011 Asia-Pacific power and energy engineering conference*. IEEE. 2011, pp. 1–4.

[133] *SAE Electric Vehicle and Plug in Hybrid Electric Vehicle Conductive Charge Coupler*. URL: `https://saemobilus.sae.org/content/J1772_201710/`.

[134] Gamal Abd El-Nasser A Said, Abeer M Mahmoud, and El-Sayed M El-Horbaty. "A comparative study of meta-heuristic algorithms for solving quadratic assignment problem". In: *arXiv preprint arXiv:1407.4863* (2014).

[135] Adella Santos et al. *Summary of travel trends: 2009 national household travel survey*. Tech. rep. United States. Federal Highway Administration, 2011.

[136] Shankar Sastry. *Nonlinear systems: analysis, stability, and control*. Vol. 10. Springer Science & Business Media, 2013.

[137] Samveg Saxena et al. "Quantifying EV battery end-of-life through analysis of travel needs with vehicle powertrain models". In: *Journal of Power Sources* 282 (2015), pp. 265–276.

[138] Linda Shapiro. *Computer vision and image processing*. Academic Press, 1992.

[139] Xinyue Shen et al. "Disciplined convex-concave programming". In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE. 2016, pp. 1009–1014.

[140] Naum Z Shor. "Quadratic optimization problems". In: *Soviet Journal of Computer and Systems Sciences* 25.6 (1987), pp. 1–11.

[141] Kenneth A Small. "A discrete choice model for ordered alternatives". In: *Econometrica: Journal of the Econometric Society* (1987), pp. 409–424.

[142] Kate A Smith. "Neural networks for combinatorial optimization: a review of more than a decade of research". In: *INFORMS Journal on Computing* 11.1 (1999), pp. 15–34.

[143] Steve Sorrell and John Dimitropoulos. "UKERC review of evidence for the rebound effect: technical report 5—energy productivity and economic growth studies". In: *UK Energy Research Centre, London* (2007).

[144] Eric Sortomme et al. "Coordinated charging of plug-in hybrid electric vehicles to minimize distribution system losses". In: *IEEE transactions on smart grid* 2.1 (2010), pp. 198–205.

[145] IBM ILOG CPLEX Optimization Studio. *CPLEX Users Manual, version 12.7*. 2017.

[146] Weijie Su, Stephen Boyd, and Emmanuel Candes. "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2510–2518.

[147] Df Tank and JJ Hopfield. "Simple'neural'optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit". In: *IEEE transactions on circuits and systems* 33.5 (1986), pp. 533–541.

[148] Gavin Taylor et al. "Training neural networks without gradients: A scalable ADMM approach". In: *International conference on machine learning*. 2016, pp. 2722–2731.

[149] Joshua Adam Taylor. *Convex optimization of power systems*. Cambridge University Press, 2015.

[150] *Tesla Q4 2020 Vehicle Production & Deliveries: Tesla Investor Relations*. URL: `https://ir.tesla.com/press-release/tesla-q4-2020-vehicle-production-deliveries`.

[151] John Tierney. "When energy efficiency sullies the environment". In: *The New York Times* 7 (2011).

[152] Bertr Travacca and Scott Moura. "Dual Hopfield methods for large-scale mixed-integer programming". In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 4959–4966.

[153] Paul Tseng. "On accelerated proximal gradient methods for convex-concave optimization". In: *submitted to SIAM Journal on Optimization* 1 (2008).

[154] Berwin A Turlach and Andreas Weingessel. "quadprog: Functions to solve quadratic programming problems". In: *R package version* (2007), pp. 1–4.

[155] *U.S. Energy Information Administration - EIA - Independent Statistics and Analysis*. URL: `https://www.eia.gov/todayinenergy/detail.php?id=46236`.

[156] Stylianos I Vagropoulos and Anastasios G Bakirtzis. "Optimal bidding strategy for electric vehicle aggregators in electricity markets". In: *IEEE Transactions on power systems* 28.4 (2013), pp. 4031–4041.

[157] Marina González Vayá and Göran Andersson. "Optimal bidding strategy of a plug-in electric vehicle aggregator in day-ahead electricity markets under uncertainty". In: *IEEE transactions on power systems* 30.5 (2014), pp. 2375–2385.

[158] Aladin Virmaux and Kevin Scaman. "Lipschitz regularity of deep neural networks: analysis and efficient estimation". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3835–3844.

[159] Robin Vujanic et al. "Vanishing duality gap in large scale mixed-integer optimization: a solution method with power system applications". In: *J. Math. Program* (2014).

[160] Haohan Wang, Bhiksha Raj, and Eric P Xing. "On the origin of deep learning". In: *arXiv preprint arXiv:1702.07800* (2017).

[161] Huaizhi Wang et al. "A review of deep learning for renewable energy forecasting". In: *Energy Conversion and Management* 198 (2019), p. 111799.

[162] Rong Long Wang, Zheng Tang, and Qi Ping Cao. "A learning method in Hopfield neural network for combinatorial optimization problem". In: *Neurocomputing* 48.1-4 (2002), pp. 1021–1024.

[163] Po-Wei Wang et al. "SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, 2019, pp. 6545–6554. URL: http://proceedings.mlr.press/v97/wang19e.html.

[164] Ryan H Wiser and Mark Bolinger. "2018 wind technologies market report". In: (2019).

[165] Maria Wolrath Söderberg and Nina Wormbs. "Grounded: Beyond flygskam". In: European Liberal Forum & Fores. 2019.

[166] XF Wu and GQ Chen. "Global primary energy use associated with production, consumption and international trade". In: *Energy Policy* 111 (2017), pp. 85–94.

[167] Xin-Yu Wu et al. "A high-performance neural network for solving linear and quadratic programming problems". In: *IEEE transactions on neural networks* 7.3 (1996), pp. 643–651.

[168] Yangyang Xu and Wotao Yin. "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion". In: *SIAM Journal on imaging sciences* 6.3 (2013), pp. 1758–1789.

[169] Ji Hoon Yoon, Ross Baldick, and Atila Novoselac. "Dynamic demand response controller based on real-time retail price for residential buildings". In: *IEEE Transactions on Smart Grid* 5.1 (2014), pp. 121–129.

[170] Jinshan Zeng et al. "Global convergence of block coordinate descent in deep learning". In: *arXiv preprint arXiv:1803.00225* (2018).

[171] Teng Zeng et al. "Inducing Human Behavior to Maximize Operation Performance at PEV Charging Station". In: *IEEE Transactions on Smart Grid* (2021), pp. 1–1. DOI: 10.1109/TSG.2021.3066998.

[172] Xinhua Zhang. *Bregman Divergence and Mirror Descent*. 2014.

[173] Ziming Zhang and Matthew Brand. "Convergent Block Coordinate Descent for Training Tikhonov Regularized Deep Neural Networks". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 1719–1728. ISBN: 978-1-5108-6096-4. URL: http://dl.acm.org/citation.cfm?id=3294771.3294935.

[174] Ahmed F Zobaa and Shady Abdel Aleem. *Uncertainties in Modern Power Systems*. Academic Press, 2020.