Large-Scale Intelligent Systems: From Network Dynamics to Optimization Algorithms

Thesis by
Navid Azizan

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2021
Defended August 25, 2020

Navid Azizan
ORCID: 0000-0002-4299-2963

ACKNOWLEDGEMENTS

I think anyone who has been to Caltech would agree that it is truly an exceptional environment, and I consider myself lucky to have had the opportunity to spend five years of my life here and interact with such an extraordinary group of scholars. So, I wish to thank numerous individuals, without whom this thesis would not have been possible.

First, and foremost, I would like to express my deepest gratitude to my two advisors, Adam Wierman and Babak Hassibi, for their generous and continuous support, guidance, and encouragement throughout my PhD, and for giving me complete freedom in pursuing my diverse interests. They have always been available for any help despite their busy schedules, and their advice extended beyond research to career planning and even personal matters. They invested a great deal of time and effort in molding me into a successful researcher and academic, and I am forever indebted to them for that. The depth and the breadth of their knowledge, their clarity of thought, their careful comments and questions, their passion for research and education, their infective energy, and their kindness have always inspired and continue to inspire me, both professionally and personally.

I am also deeply indebted to the chair of my thesis committee, Steven Low, who in many ways acted as another advisor for me, and was in no small part responsible for my transformative experience at Caltech. I am grateful for his always thoughtful and insightful comments and for his unwavering willingness to help. I am also very grateful to Yisong Yue for his support and guidance, his valuable feedback on my job talk, for giving me the opportunity to supervise student projects in his class, and for serving on my thesis committee.

Beyond my committee, I am very grateful to other Caltech faculty members, especially Venkat Chandrasekaran, K. Mani Chandy, Richard M. Murray, John Doyle, Joel A. Tropp, Thomas Vidick, Anima Anandkumar, Leonard Schulman, Houman Owhadi, Pietro Perona, Azita Emami, and Ali Hajimiri, for guidance and for providing a stimulating environment for my intellectual growth.

The work in this thesis is the result of many collaborations. I would like to thank my wonderful collaborators and friends, Yu Su, Sahin Lale, Christos Thrampoulidis, Niangjun Chen, Wael Halbawi, Krishnamurthy Dvijotham, Linqi Guo, and Farshad Lahouti. Special thanks are also due to the CMS administrative staff, especially

Maria Lopez, Sheila Shull, Sydney Garstang, and Christine Ortega, for taking care of all the administrative issues seamlessly so that I could focus on the work presented in this thesis.

Finally, this thesis is dedicated to my parents and my brother, for all the years of love and support.

ABSTRACT

The expansion of large-scale technological systems such as electrical grids, transportation networks, health care systems, telecommunication networks, the Internet (of things), and other societal networks has created numerous challenges and opportunities at the same time. These systems are often not yet as robust, efficient, sustainable, or smart as we would want them to be. Fueled by the massive amounts of data generated by all these systems, and with the recent advances in making sense out of data, there is a strong desire to make them more intelligent. However, developing *large-scale intelligent systems* is a multifaceted problem, involving several major challenges. First, large-scale systems typically exhibit *complex dynamics* due to the large number of entities interacting over a network. Second, because the system is composed of many interacting entities, that make decentralized (and often self-interested) decisions, one has to properly design *incentives and markets* for such systems. Third, the massive computational needs caused by the scale of the system necessitate performing computations in a *distributed* fashion, which in turn requires devising new algorithms. Finally, one has to create algorithms that can *learn from* the copious amounts of data and generalize well. This thesis makes several contributions related to each of these four challenges.

Analyzing and understanding the network dynamics exhibited in societal systems is crucial for developing systems that are robust and efficient. In Part I of this thesis, we study one of the most important families of network dynamics, namely, that of *epidemics*, or *spreading processes*. Studying such processes is relevant for understanding and controlling the spread of, e.g., contagious diseases among people, ideas or fake news in online social networks, computer viruses in computer networks, or cascading failures in societal networks. We establish several results on the exact Markov chain model and the nonlinear "mean-field" approximations for various kinds of epidemics (i.e., SIS, SIRS, SEIRS, SIV, SEIV, and their variants).

Designing incentives and markets for large-scale systems is critical for their efficient operation and ensuring an alignment between the agents' decentralized decisions and the global goals of the system. To that end, in Part II of this thesis, we study these issues in markets with *non-convex* costs as well as *networked* markets, which are of vital importance for, e.g., the smart grid. We propose novel pricing schemes for such markets, which satisfy all the desired market properties. We also reveal issues in the current incentives for distributed energy resources, such as renewables, and design

optimization algorithms for efficient management of aggregators of such resources.

With the growing amounts of data generated by large-scale systems, and the fact that the data may already be dispersed across many units, it is becoming increasingly necessary to run computational tasks in a distributed fashion. Part III concerns developing algorithms for distributed computation. We propose a novel consensus-based algorithm for the task of solving large-scale *systems of linear equations*, which is one of the most fundamental problems in linear algebra, and a key step at the heart of many algorithms in scientific computing, machine learning, and beyond. In addition, in order to deal with the issue of heterogeneous delays in distributed computation, caused by slow machines, we develop a new *coded computation* technique. In both cases, the proposed methods offer significant speed-ups relative to the existing approaches.

Over the past decade, *deep learning* methods have become the most successful learning algorithms in a wide variety of tasks. However, the reasons behind their success (as well as their failures in some respects) are largely unexplained. It is widely believed that the success of deep learning is not just due to the deep architecture of the models, but also due to the behavior of the optimization algorithms, such as stochastic gradient descent (SGD), used for training them. In Part IV of this thesis, we characterize several properties, such as minimax optimality and implicit regularization, of SGD, and more generally, of the family of *stochastic mirror descent (SMD)*. While SGD performs an implicit regularization, this regularization can be effectively controlled using SMD with a proper choice of mirror, which in turn can improve the generalization error.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Navid Azizan. "Optimization Algorithms for Large-Scale Systems: From Deep Learning to Energy Markets". In: *SIGMETRICS Performance Evaluation Review* 47.3 (2020), pp. 2–5. ISSN: 0163-5999. DOI: 10.1145/3380908.3380910.

[2] Navid Azizan et al. "A Study of Generalization of Stochastic Mirror Descent Algorithms on Overparameterized Nonlinear Models". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 3132–3136. DOI: 10.1109/ICASSP40776.2020.9053864.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[3] Navid Azizan et al. "Optimal Pricing in Markets with Nonconvex Costs". In: *Operations Research* 68.2 (2020), pp. 480–496. DOI: 10.1287/opre.2019.1900.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[4] Navid Azizan and Babak Hassibi. "A Characterization of Stochastic Mirror Descent Algorithms and their Convergence Properties". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 5167–5171. DOI: 10.1109/ICASSP.2019.8682271.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[5] Navid Azizan and Babak Hassibi. "A Stochastic Interpretation of Stochastic Mirror Descent: Risk-Sensitive Optimality". In: *2019 58th IEEE Conference on Decision and Control (CDC)*. 2019, pp. 3960–3965. DOI: 10.1109/CDC40024.2019.9030229.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[6] Navid Azizan and Babak Hassibi. "Stochastic gradient/mirror descent: Minimax optimality and implicit regularization". In: *2019 International Conference on Learning Representations (ICLR)*. 2019.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[7] Navid Azizan et al. "Distributed Solution of Large-Scale Linear Systems via Accelerated Projection-Based Consensus". In: *IEEE Transactions on Signal Processing* 67.14 (2019), pp. 3806–3817. DOI: 10.1109/TSP.2019.2917855.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[8] Navid Azizan et al. “Optimal Pricing in Markets with Non-Convex Costs”. In: *Proceedings of the 2019 ACM Conference on Economics and Computation (EC)*. Phoenix, AZ, USA, 2019, p. 595. ISBN: 978-1-4503-6792-9. DOI: 10.1145/3328526.3329575.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[9] Navid Azizan et al. “Stochastic Mirror Descent on Overparameterized Nonlinear Models: Convergence, Implicit Regularization, and Generalization”. In: *2019 International Conference on Machine Learning (ICML) Generalization Workshop*. 2019.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[10] Navid Azizan and Babak Hassibi. “Stochastic gradient/mirror descent: Minimax optimality and implicit regularization”. In: *2018 Neural Information Processing Systems (NeurIPS) Deep Learning Theory Workshop*. 2018.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[11] Navid Azizan et al. “Distributed Solution of Large-Scale Linear Systems via Accelerated Projection-Based Consensus”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 6358–6362. DOI: 10.1109/ICASSP.2018.8462630.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[12] Navid Azizan et al. “Opportunities for Price Manipulation by Aggregators in Electricity Markets”. In: *IEEE Transactions on Smart Grid* 9.6 (2018), pp. 5687–5698. DOI: 10.1109/TSG.2017.2694043.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[13] Wael Halbawi et al. “Improving Distributed Gradient Descent Using Reed-Solomon Codes”. In: *2018 IEEE International Symposium on Information Theory (ISIT)*. 2018, pp. 2027–2031. DOI: 10.1109/ISIT.2018.8437467.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[14] Navid Azizan et al. “Analysis of exact and approximated epidemic models over complex networks”. In: *arXiv preprint arXiv:1609.09565* (2016). URL: http://arxiv.org/abs/1609.09565.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[15] Navid Azizan et al. “Improved bounds on the epidemic threshold of exact SIS models on complex networks”. In: *2016 55th IEEE Conference on Decision and Control (CDC)*. 2016, pp. 3560–3565. DOI: 10.1109/CDC.2016.7798804.

N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[16] Navid Azizan et al. “Opportunities for Price Manipulation by Aggregators in Electricity Markets”. In: *SIGMETRICS Performance Evaluation Review* 44.2 (2016), pp. 49–51. ISSN: 0163-5999. DOI: 10.1145/3003977.3003995.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

[17] Navid Azizan and Babak Hassibi. “SIRS epidemics on complex networks: Concurrence of exact Markov chain and approximated models”. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015, pp. 2919–2926. DOI: 10.1109/CDC.2015.7402660.
N.A. contributed to the conception of the project, designing the methods, performing the analysis, and writing the manuscript.

TABLE OF CONTENTS

Acknowledgements . . . iii
Abstract . . . v
Published Content and Contributions . . . vii
Table of Contents . . . ix
List of Illustrations . . . xiii
List of Tables . . . xviii
Chapter I: Introduction . . . 1
1.1 Major Challenges . . . 1
1.2 Synopsis of Part I: Network Dynamics . . . 2
1.3 Synopsis of Part II: Incentives and Markets . . . 4
1.4 Synopsis of Part III: Distributed Computation . . . 5
1.5 Synopsis of Part IV: Learning from Data . . . 7

I Network Dynamics 10
Chapter II: Epidemics over Complex Networks: Analysis of Exact and Approximate Models . . . 11
2.1 Introduction . . . 12
2.2 Models . . . 14
2.3 Results on the Nonlinear MFA Model . . . 22
2.4 Results on the Exact Markov Chain Model . . . 26
2.5 Heterogeneous Network Models . . . 29
2.6 Pairwise and Higher-Order Approximate Models . . . 30
2.7 Summary and Conclusion . . . 30
2.A Additional Models . . . 33
Chapter III: Improved Bounds on the Epidemic Threshold of the Exact Models 40
3.1 Introduction . . . 40
3.2 The Markov Chain and Marginal Probabilities of Infection . . . 42
3.3 Pairwise Probabilities (p_{ij}) . . . 46
3.4 An Alternative Pairwise Probability (q_{ij}) . . . 49
3.5 Experimental Results . . . 51
3.6 Conclusion and Future Work . . . 53

II Incentives and Markets 55
Chapter IV: Optimal Pricing in Markets with Non-Convex Costs . . . 56
4.1 Introduction . . . 56
4.2 Market Description and Pricing Objectives . . . 59
4.3 Proposed Scheme: Equilibrium-Constrained Pricing . . . 62

4.4 Equilibrium-Constrained Pricing for Networked Markets 73
4.5 Existing Pricing Schemes 79
4.6 Experimental Results 84
4.7 Concluding Remarks 89
4.A Supplement to Section 4.3.1 91
4.B Supplement to Section 4.3.2 94
4.C Supplement to Section 4.4 97
Chapter V: Managing Aggregators in the Smart Grid 99
5.1 Introduction 99
5.2 System Model 103
5.3 The Market Behavior of the Aggregator 105
5.4 The Impact of Strategic Curtailment 108
5.5 Optimizing Curtailment Profit 111
5.6 Concluding Remarks 117
5.A Connections between Curtailment Profit and Market Power 119
5.B Proof of Lemma 25 (Monotonicity of LMP) 121
5.C Proof of Theorem 26 (Exact Single-Bus) 123
5.D Proof of Theorem 27 (Approximate Multi-Bus) 125

III Distributed Computation 128
Chapter VI: Distributed Solution of Large-Scale Systems of Equations 129
6.1 Introduction 130
6.2 The Setup 131
6.3 Accelerated Projection-Based Consensus 132
6.4 Comparison with Related Methods 139
6.5 Underdetermined System 144
6.6 Experimental Results 150
6.7 A Distributed Preconditioning to Improve Gradient-Based Methods . 152
6.8 Conclusion 153
Chapter VII: Coded Computation for Distributed Gradient Descent 154
7.1 Introduction 154
7.2 Preliminaries 157
7.3 Code Construction 160
7.4 Efficient Online Decoding 166
7.5 Analysis of Total Computation Time 168
7.6 Numerical Results 171
7.7 Conclusion 173

IV Learning from Data 174
Chapter VIII: Minimax Optimality and Implicit Regularization of Stochastic Gradient/Mirror Descent 175
8.1 Introduction 175
8.2 Preliminaries 177

8.3 Warm-up: Revisiting SGD on Square Loss of Linear Models 178
8.4 Main Result: General Characterization of Stochastic Mirror Descent 183
8.5 Convergence and Implicit Regularization in Over-Parameterized Models 186
8.6 Concluding Remarks 189
8.A Proof of Lemma 41 (Fundamental Identity) 190
8.B Proof of Theorem 43 (Minimax Optimality) 192
8.C Proof of Proposition 46 (Convergence) 195
8.D Time-Varying Step-Size 197
Chapter IX: SMD on Overparameterized Nonlinear Models 199
9.1 Introduction 200
9.2 Background 202
9.3 Training Deep Neural Networks with SMD 203
9.4 Theoretical Results 206
9.5 Related Work 210
9.6 Experimental Results 212
9.7 Conclusion 214
9.A Proofs of the Theoretical Results 215
9.B More Details on the Experimental Results 223
Chapter X: Stochastic Results: Risk-Sensitive Optimality and Mean-Square Convergence of SMD 237
10.1 Introduction 237
10.2 Background 238
10.3 Risk-Sensitive Optimality of SMD 245
10.4 Symmetric SMD (SSMD) 249
10.5 Mean-Square Convergence of SMD 249
10.6 Conclusion 251
Bibliography 252

LIST OF ILLUSTRATIONS

Number *Page*

2.1 State diagram of a single node in different models. Wavy arrows represent exogenous (network-based) transition. S stands for susceptible (healthy), E for exposed, I for infected/infectious, and R for recovered. 15

2.2 Reduced Markov chain of a single node in the steady state. 21

2.3 Summary of known results for different models. The results have been illustrated as a function of $\frac{\beta\lambda_{max}(A)}{\delta}$. MC stands for the Markov chain model. MFA stands for the mean-field approximation (the nonlinear model). 23

2.4 A typical example of the evolution of an SIS epidemic over an Erdős-Rényi graph with $n = 2000$ nodes and $\lambda_{\max}(A) = 16.159$. When the condition $\frac{\beta\lambda_{\max}(A)}{\delta} < 1$ is satisfied (e.g., $\beta = 0.055$, $\delta = 0.9$) the epidemic decays exponentially, and dies out quickly (blue curve). In contrast, when $\frac{\beta\lambda_{\max}(A)}{\delta} > 1$ (e.g., $\beta = 0.056$, $\delta = 0.9$), the epidemic does not exhibit convergence to the disease-free state in any observable time (red curve). In fact, the epidemic keeps spreading around the nontrivial fixed point. 28

2.5 The evolution of (a) SIS/SIRS/SEIRS, (b) SIV/SEIV (infection-dominant), (c) SIV/SIEV (vaccination-dominant) epidemics over an Erdős-Rényi graph with $n = 2000$ nodes. The blue curves show fast extinction of the epidemic. The red curves show epidemic spread around the nontrivial fixed point. 31

3.1 State diagram of a single node in the SIS model, and the transition rates. Wavy arrow represents exogenous (neighbor-based) transition. β : probability of infection per infected link, δ : probability of recovery. 42

3.2 Evolution of the SIS epidemic over a star graph with $n = 2000$ nodes, with two values of $\rho(M')$ below and above 1. When $\rho(M') = 0.99 < 1$, we observe fast extinction of the epidemic (blue curve). The condition also seems very tight, as for $\rho(M') = 1.05 > 1$, the epidemic does not die out (red curve). This is while the previously known bound is not informative at all ($\frac{\beta\lambda_{\max}(A)}{\delta} = 1.93 > 1$ for the first case, and $2.33 > 1$ for the second one). 53

4.1 An illustration of the set Λ for an example with 3 non-convex cost functions. The three blue curves are the cost functions. The (dashed and solid) red lines lie below all the cost functions and their slopes are in Λ. The (slope of the) solid red line corresponds to the largest element of Λ. 68
4.2 An example of the binary tree defined by Algorithm 1 for $n = 8$. The faded circles correspond to the added dummy nodes. 71
4.3 An illustration of shadow pricing for the case of 3 convex cost functions. The points indicated by $*$ show the optimal quantities. The 3 functions have the same derivative at their optimal quantities, and the tangent line lies below the function (because of convexity). The red (solid) line that passes through the origin is the uniform price function, which is parallel to the three lines. 81
4.4 An example with cost functions of the form of linear plus startup cost. 86
4.5 An example with cost functions of the form of quadratic plus startup cost. 87
4.6 A schematic drawing for two connected markets with a constraint on flow capacity. 88
4.7 An example of two connected markets with a constraint on the flow capacity. 89
4.8 The transformation of an arbitrary-degree tree to a binary tree. 97
5.1 The 6-bus example network from [35], used to illustrate the effect of curtailment. 108
5.2 The locational marginal prices for the 6-bus example before and after the curtailment. 109
5.3 The profit under the normal (no-curtailment) condition and under (optimal) strategic curtailment, as a function of size of the aggregator in IEEE test case networks: a) IEEE 14-Bus Case, b) IEEE 30-Bus Case, and c) IEEE 57-Bus Case. The difference between the two curves is the curtailment profit. 110
5.4 A heat map of the impact of coordinated curtailment on the prices in the IEEE 14-bus network. Aggregator nodes are 2, 7, 10, and 14. . . 111
5.5 The LMP at bus i as a function of curtailed generation at that bus. Shaded areas indicate the aggregator's revenue at the normal condition and at the curtailment. 113

5.6 The representation of a binary tree. For any node i, and its children denoted $c_1(i), c_2(i)$. 116
5.7 The 9-bus acyclic network from [116], used for the evaluation of the proposed approximation algorithm. 117
5.8 The difference from the optimal solution as a function of the running time of the algorithm, in the 9-bus network with 1% curtailment allowance. 118
6.1 Schematic representation of the taskmaster and the m machines/cores. Each machine i has only a subset of the equations, i.e., $[A_i, b_i]$. . . . 132
6.2 The decay of the error for different distributed algorithms, on two real problems from Matrix Market [141] (QC324: Model of H_2^+ in an Electromagnetic Field, and ORSIRR 1: Oil reservoir simulation). n = # of variables, N = # of equations, m = # of workers, p = # of equations per worker. 151
7.1 Schematic representation of the taskmaster and the n workers. 157
7.2 This plot corresponds to a setup where the number of training examples is $N = 12000$ and $c_g = 3 \times 10^{-6}$ to give $Nc_g = 0.035$. The parameters of the Pareto distribution corresponding to the delay is characterized by $t_0 = 0.001$ and $\xi = 1.1$. The optimizer of this function as predicted by (7.26) is $\alpha^* = 0.1477$. This point is denote by the star symbol. . . 171
7.3 The comparison between the test error of different schemes as a function of time, for a softmax regression model trained using distributed gradient descent on $n = 80$ machines. The model was trained on 12000 examples from the MNIST database [121] and validated on a test set of size 10000. The Reed–Solomon based scheme (Coded - RS) waits for $f_{\text{RS}} = 68$ machines, while the one corresponding to [197] (Coded - MDS) waits for $f_{\text{MDS}} = 33$. f_{RS} and f_{MDS} were obtained by numerically optimizing (7.21). The two coded schemes outperform the uncoded ones. Coded-RS denotes the proposed scheme. 172
8.1 Illustration of Lemma 38. Each step of SGD can be viewed as a transformation of the uncertainties with the right coefficients. 180
8.2 The training loss and actual error of stochastic mirror descent for compressed sensing. SMD recovers the actual sparse signal. 188

9.1 Generalization performance of different SMD algorithms on the CIFAR-10 dataset using ResNet-18. ℓ_{10} performs consistently better, while ℓ_1 performs consistently worse. The red line shows the state of the art on ResNet-18 for CIFAR-10 (93.02%)[135]. 204
9.2 Histogram of the absolute value of the final weights in the network for different SMD algorithm with different potentials. Note that each of the four histograms corresponds to an 11×10^6-dimensional weight vector that perfectly interpolates the data. Even though the weights remain quite small, the histograms are drastically different. ℓ_1-SMD induces sparsity on the weights. SGD appears to lead to a Gaussian distribution on the weights. ℓ_3-SMD starts to reduce the sparsity, and ℓ_{10} shifts the distribution of the weights significantly, so much so that almost all the weights are non-zero. 205
9.3 An illustration of the parameter space. $\mathcal{W}$ represents the set of global minima, w_0 is the initialization, $\mathcal{B}$ is the local neighborhood, w^* is the closest global minimum to w_0 (in Bregman divergence), and w_∞ is the minimum that SMD converges to. 207
9.4 An illustration of $D_{L_i}(w, w') \geq 0$ in a local region in Assumption 1. . 208
9.5 An illustration of the experiments in Table 9.1. 210
9.6 An illustration of the experiments in Table 9.2. 213
9.7 Distances between a particular initial point and all the final points obtained by both different initializations and different mirrors. The smallest distance, among all initializations and all mirrors, corresponds exactly to the final point obtained from that initial point by SGD. This trend is observed consistently for all other mirror descents and all initializations (see the results in Tables 9.8 and 9.9 in the appendix). . 213
9.8 Different Bregman divergences between all the final points and all the initial points for different mirrors in MNIST dataset using a standard CNN. Note that the smallest element in every single row is on the diagonal, which confirms the theoretical results. 227
9.9 Different Bregman divergences between all the final points and all the initial points for different mirrors in CIFAR-10 dataset using ResNet-18. Note that the smallest element in every single row is on the diagonal, which confirms the theoretical results. 232

9.10 An illustration of the experimental results. For each initialization w_0, we ran different SMD algorithms until convergence to a point on the set $\mathcal{W}$ (zero training error). We then measured all the pairwise distances from different w_∞ to different w_0, in different Bregman divergences. The closest point (among all initializations and all mirrors) to any particular initialization w_0 in Bregman divergence with potential $\psi(\cdot) = \|\cdot\|_q^q$ is exactly the point obtained by running SMD with potential $\|\cdot\|_q^q$ from w_0. 233
9.11 Histogram of the absolute value of the initial weights in the network (half-normal distribution). 234
9.12 Histogram of the absolute value of the final weights in the network for different SMD algorithms: (a) ℓ_1-SMD, (b) ℓ_2-SMD (SGD), (c) ℓ_3-SMD, and (d) ℓ_{10}-SMD. Note that each of the four histograms corresponds to an 11×10^6-dimensional weight vector that perfectly interpolates the data. Even though the weights remain quite small, the histograms are drastically different. ℓ_1-SMD induces sparsity on the weights, as expected. SGD does not seem to change the distribution of the weights significantly. ℓ_3-SMD starts to reduce the sparsity, and ℓ_{10} shifts the distribution of the weights significantly, so much so that almost all the weights are non-zero. 235
9.13 Generalization performance of different SMD algorithms on the CIFAR-10 dataset using ResNet-18. ℓ_{10} performs consistently better, while ℓ_1 performs consistently worse. 236
10.1 Bregman divergence. 241
10.2 Local Conservation Law of SMD. 243
10.3 w_∞ is the closest solution (among all solutions $\mathcal{W}$) to w_0. Note that this picture is only for the Euclidean distance; in general the "closest" is measured in Bregman divergence. 245

LIST OF TABLES

Number *Page*

3.1 Performance of the proposed bounds M' and M'' in comparison with the previous bound M. Boldface values show an improvement over M. The signs next to $\rho(M')$ indicate whether the non-negativity condition (required for the proof) holds. 52

4.1 Summary of common pricing schemes and their properties. 82

4.2 Summary of the production characteristics in the modified Scarf's example. 85

4.3 Summary of the new cost functions in the modified Scarf's example. . 86

6.1 A summary of the convergence rates of different methods. DGD: Distributed Gradient Descent, D-NAG: Distributed Nesterov's Accelerated Gradient Descent, D-HBM: Distributed Heavy-Ball Method, Mou et al: Consensus algorithm of [146], B-Cimmino: Block Cimmino Method, APC: Accelerated Projection-based Consensus. The smaller the convergence rate is, the faster is the method. Note that $\rho_{\text{GD}} \geq \rho_{\text{NAG}} \geq \rho_{\text{HBM}}$ and $\rho_{\text{Mou}} \geq \rho_{\text{Cim}} \geq \rho_{\text{APC}}$. 139

6.2 A comparison between the condition numbers of $A^T A$ and X for some examples. m is the number of machines/partitions. The condition number of X is typically much smaller (better). Remarkably, the difference is even more pronounced when A has non-zero mean. . . . 145

6.3 A comparison between the optimal convergence time T ($= \frac{1}{-\log \rho}$) of different methods on real and synthetic examples. Boldface values show the smallest convergence time. QC324: Model of H_2^+ in an Electromagnetic Field. ORSIRR 1: Oil Reservoir Simulation. ASH608: Original Harwell sparse matrix test collection. 146

9.1 Fixed Initialization. Distances from final points (global minima) obtained by different algorithms (columns) from the same initialization (Fig. 9.5), measured in different Bregman divergences (rows). First Row: The closest point to w_0 in ℓ_1 Bregman divergence, among the four final points, is exactly the one obtained by SMD with 1-norm potential. Second Row: The closest point to w_0 in ℓ_2 Bregman divergence (Euclidean distance), among the four final points, is exactly the one obtained by SGD. Third Row: The closest point to w_0 in ℓ_3 Bregman divergence, among the four final points, is exactly the one obtained by SMD with 3-norm potential. Fourth Row: The closest point to w_0 in ℓ_{10} Bregman divergence, among the four final points, is exactly the one obtained by SMD with 10-norm potential. 211
9.2 Fixed Mirror: SGD. Pairwise distances between different initial points and the final points obtained from them by SGD (Fig. 9.6). Row i: The closest final point to the initial point i, among all the eight final points, is exactly the one obtained by the algorithm from initialization i. 212
9.3 MNIST Initial Point 1. 224
9.4 MNIST Initial Point 2. 224
9.5 MNIST Initial Point 3. 224
9.6 MNIST Initial Point 4. 225
9.7 MNIST Initial Point 5. 225
9.8 MNIST Initial Point 6. 225
9.9 MNIST 1-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 1-norm. 226
9.10 MNIST 2-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 2-norm (SGD). 226
9.11 MNIST 3-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 3-norm. 226
9.12 MNIST 10-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 10-norm. 226
9.13 CIFAR-10 Initial Point 1. 228
9.14 CIFAR-10 Initial Point 2. 228
9.15 CIFAR-10 Initial Point 3. 228
9.16 CIFAR-10 Initial Point 4. 229
9.17 CIFAR-10 Initial Point 5. 229
9.18 CIFAR-10 Initial Point 6. 229

9.19 CIFAR-10 Initial Point 7. 229
9.20 CIFAR-10 Initial Point 8. 229
9.21 CIFAR-10 1-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 1-norm. 230
9.22 CIFAR-10 2-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 2-norm (SGD). 230
9.23 CIFAR-10 3-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 3-norm. 230
9.24 CIFAR-10 10-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 10-norm. 231

Chapter 1

INTRODUCTION

Our technological systems are arguably at the dawn of a major transformation. We have built complex systems such as electrical grids, transportation networks, health care systems, telecommunication networks, the Internet (of things), and other societal networks, which have enabled connecting large numbers of entities or people. While immensely helpful, these systems are, in many senses, not yet as *robust*, *efficient*, *sustainable*, or *smart* as we would want them to be. However, that is beginning to change. The formation of these large-scale systems, while posing enormous challenges (such as how to manage them efficiently), has created tremendous opportunities for developing "more intelligent" systems. With the massive amounts of data generated by all these systems, and with the major advances during the recent years in areas such as machine learning and data science, network science, and market design, we are at a unique time in history to revolutionize these systems and pave the way for the development of what can be referred to as *large-scale intelligent systems*. This thesis is broadly aimed at addressing some of the key challenges towards realizing this goal, and laying a foundation for analyzing and designing such systems.

1.1 Major Challenges

Developing large-scale intelligent systems is a multifaceted problem and requires a confluence of disciplines. In particular, over the past few decade, we have seen remarkable progress in this interdisciplinary endeavor from various fields such as networks science, machine learning, statistics, optimization, control theory, and game theory, among others. Despite this progress, we are still far from realizing that vision. Some of the key challenges in designing large-scale intelligent systems can be summarized as follows.

1. Complex Dynamics: Large-scale systems typically exhibit complex dynamics, caused by the large number of entities interacting with one another, often over a network. Analyzing and understanding these dynamics is crucial for creating systems that are efficient and robust.

2. Incentives and Markets: Because the system is composed of a large number

of interacting entities, that make decisions in a decentralized (and often self-interested) manner, it is critical to design incentives and markets for the system in such a way that ensures an alignment of the agents' decisions and the overall goals of the system.

3. Distributed Computation: The massive computational needs due to the scale of the system (and the fact that the data may be dispersed across many entities) make it virtually impossible to carry out computations at a central unit. Therefore, devising algorithms that can run in a distributed or parallel fashion is of vital importance for such systems.

4. Learning from Data: Lastly, an important aspect of an intelligent system is the ability to learn from data, and the enormous amount of data generated by these large-scale systems makes them uniquely appealing for this purpose. Creating learning algorithms that can generalize well is an ongoing enterprise, with notable successes and many unsolved problems.

Some of the common lower-level obstacles that often arise in addressing the above challenges are those of *networks* and/or *non-convexities*. These issues, as will be discussed later, complicate both understanding the behavior of these systems as well as designing suitable algorithms for them.

This thesis is organized into four main parts, based on the four major challenges discussed above. While related, the four parts need not be read in order (or in their entirety) when reading this thesis. To further allow for a modular reading, each chapter is aimed to be self-contained.[1] In what follows, we summarize the main contributions of the thesis in each part.

1.2 Synopsis of Part I: Network Dynamics

As mentioned earlier, understanding and analyzing the dynamics of networks is crucial for developing societal systems that are efficient and robust. In Part I, we study one of the most prominent families of network dynamics, namely, that of *spreading processes*, or *epidemics*. Studying such processes is of great importance for understanding and controlling how, e.g., contagious diseases spread among humans, ideas or fake news spread in online social networks, cascading failures happen in power networks, or computer viruses spread in computer networks, and thus has

[1] In doing so, some minor redundancy across chapters has been introduced.

applications in many areas such as epidemiology [29], information propagation [109, 56], viral marketing [168, 175], and network security [8, 2].

We consider the spread of discrete-time epidemics over arbitrary networks for well-known propagation models, namely SIS (susceptible-infected-susceptible), SIRS (susceptible-infected-recovered-susceptible), SEIRS (susceptible-exposed-infected-recovered-susceptible), SIV (susceptible-infected-vaccinated), SEIV (susceptible-exposed-infected-vaccinated), and their variants. Such spreading processes can be normally described by Markov chains with an exponential number of states in the number of nodes. Since analyzing these Markov chain models is complicated, various linear and nonlinear lower-dimensional approximations of them have been proposed and studied in the literature. The most common of these is the nonlinear "mean-field" approximation and its linearization around the disease-free fixed point, whose number of states are linear in the number of nodes.

In **Chapter 2**, we provide a *complete global analysis* of the epidemic dynamics of the *nonlinear mean-field approximation*, as well as a sufficient condition for *fast extinction of the epidemic* in the *exact Markov chain model*, for the aforementioned propagation models (SIS, SIRS, SEIRS, SIV, and SEIV). In particular, we show that for most propagation models, the global dynamics of the nonlinear model coincides with the stability of the linear model, and takes on one of two forms: either the epidemic dies out, or it converges to another unique fixed point (the so-called endemic state where a constant fraction of the nodes remain infected). We tie in these results with the exact Markov chain model by showing that the linear model provides an upper-bound on the true marginal probabilities of infection, and that this is the *tightest upper-bound* that involves only marginals, in the "low-infection" regime. This bound implies that under the specific threshold where the disease-free state is a globally-stable fixed point of the mean-field model, the exact underlying Markov chain has a sublinear mixing time, which means the epidemic dies out quickly.

The threshold condition for fast mixing of the Markov chain has been shown not to be tight in several cases, such as in a star network. In **Chapter 3**, we provide tighter upper bounds on the exact marginal probabilities of infection, by also taking *pairwise infection probabilities* into account. Based on this improved bound, we derive tighter eigenvalue conditions that guarantee fast mixing (i.e., logarithmic mixing time) of the chain. Comparisons between the new condition and the known one on various networks with various epidemic parameters demonstrates significant improvement of the threshold condition.

1.3 Synopsis of Part II: Incentives and Markets

As discussed earlier, a critical aspect of developing large-scale intelligent systems is designing incentives and markets, in such a way that ensures efficient operation of the system and effective management of the available resources. One of the key challenges that arises in such markets (and is of critical importance for, e.g., energy markets) is that of *non-convexities*. Non-convexities in cost functions arise due to start-up or shut-down costs, indivisibilities, avoidable costs, or simply economies of scale, and there may be no linear prices that support a competitive market equilibrium in their presence[46, 83]. Another important challenge in these markets is that there are *network* constraints that have to be taken into account.

Despite the large body of work on the pricing problem (especially during the past decade, motivated by the deregulation of the electricity markets in the US and around the world), the existing schemes have several shortcomings. Most of the existing schemes are proposed for specific classes of non-convex cost functions, and cannot handle more general non-convexities. Furthermore, even the ones that are applicable for general cost functions fail to satisfy some of the key desired properties of the market, such as economic efficiency or supporting a competitive equilibrium. In addition, none of the existing schemes is accompanied by a computationally tractable algorithm for general non-convex costs.

In **Chapter 4**, we propose a pricing scheme called *Equilibrium-Constrained (EC) pricing* for markets with general non-convex costs that designs arbitrary parametric price functions and addresses all the aforementioned issues. Optimizing simultaneously for the quantities (allocations) and the price parameters allows our scheme to find prices that are typically economically more efficient. Further, the ability to use arbitrarily specified parametric price functions (e.g., piece-wise linear, quadratic, etc.) enables our approach to design price functions that are less discriminatory, while still supporting a competitive equilibrium. Further, our pricing scheme is accompanied by a computationally efficient (polynomial-time) approximation algorithm which allows one to find the approximately-optimal schedule and prices for general non-convex cost functions. The proposed framework applies to the case of networked markets as well, which, to the best of our knowledge, had not been considered in previous work.

Increasing the penetration of distributed, renewable energy resources into the electricity grid is a crucial part of building a sustainable energy landscape, and the entities that have been most successful at this are *aggregators*, e.g., SolarCity, Tesla, Enphase, Sunnova, SunPower, and ChargePoint. Aggregators play a variety of

important roles in the construction of a sustainable grid: (1) they are on the front lines of the battle to promote widespread adoption of distributed energy resources by households and businesses, and (2) they provide a single interface point where utilities and Independent System Operators (ISOs) can interact with a fleet of distributed energy resources across the network in order to obtain a variety of services, from renewable generation capacity to demand response. However, in addition to the benefits they provide, aggregators also create new challenges. On the side of the aggregator, the management of a geographically diverse fleet of distributed energy resources is a difficult algorithmic challenge. On the side of the operator, the participation of aggregators in electricity markets presents unique challenges in terms of monitoring and mitigating the potential of exercising market power. In particular, unlike traditional generation resources, the ISO cannot verify the availability of the generation resources of aggregators, and this creates significant opportunities for the aggregators to manipulate prices through strategic curtailment of the resources.

In **Chapter 5**, we address both the algorithmic challenge of managing an aggregator and the economic challenge of measuring the potential for an aggregator to manipulate prices. Specifically, we provide a new algorithmic framework for managing the participation of an aggregator in electricity markets, and use this framework to evaluate the potential for aggregators to exercise market power. To those ends, we make three main contributions. First, we introduce a *new model for studying the market behavior of aggregators* of distributed generation in the real-time market. Second, we *quantify opportunities for price manipulation* by the aggregators. Our results reveal that, in practical scenarios, strategic curtailment can have a significant impact on prices, and yield much higher profits for the aggregators. In particular, the prices can be impacted up to a few tens of $/MWh in some cases, and there is often more than 25% higher profit, even with curtailments limited to 1%. Third, we provide a *novel approach for managing the participation* of an aggregator in the market. The problem is NP-hard in general and is a bilevel quadratic program, which is notoriously difficult in practice. However, we develop an efficient algorithm for aggregators in radial networks which can be used by the aggregator to approximate the optimal allocation strategy and also by the operator to assess the potential for strategic curtailment.

1.4 Synopsis of Part III: Distributed Computation

Distributed computation is an integral part of large-scale intelligent systems. With the growing size of datasets, due to high computational and/or memory requirements,

it is increasingly necessary to run the tasks in a distributed fashion. For this reason, parallel and distributed computation has attracted a lot of attention in recent years for large-scale computing applications, such as for machine learning [44, 173, 228, 77]. In order to devise efficient distributed algorithms, one has to address a number of key questions such as: *What computation should each processor carry out?*, *What messages should be communicated between the processors and the taskmaster?*, *How does the distributed implementation fare in terms of computational complexity?*, *What is the rate of convergence in the case of iterative algorithms?*, and *How to handle delays and straggling workers?*

One of the most fundamental problems in linear algebra, which also a key step at the heart of many algorithms in optimization, machine learning, scientific computing, and beyond, is that of solving a *large-scale system of linear equations*. In **Chapter 6**, we consider a common scenario in which a taskmaster intends to solve a large-scale system of linear equations by distributing subsets of the equations among a number of computing machines/cores. We propose a novel algorithm called *Accelerated Projection-based Consensus (APC)* for solving this problem. While each machine can easily find "a" solution to its own underdetermined problem, the overall solution should be a solution to every machine's problem. The idea is based on a carefully-constructed consensus, which ensures that each machine's variable remains a solution to its problem, while moving towards the solutions of the other machines. The convergence behavior of the proposed algorithm is analyzed in detail and is *analytically shown to compare favorably* with the convergence rate of alternative distributed methods, namely distributed gradient descent, distributed versions of Nesterov's accelerated gradient descent and heavy-ball method, the block Cimmino method, and ADMM. On randomly chosen linear systems, as well as on real-world data sets, the proposed method offers *order-of-magnitude speed-up* relative to the aforementioned methods.

When a task is divided among a number of machines, while the *computation time* of each machine is significantly reduced, the taskmaster has to wait for all the machines in order to be able to recover the desired computation. One issue faced in practice is the delay incurred due to the presence of slow machines, known as *stragglers*. In the face of substantial or heterogeneous delays, distributed computing may suffer from being slow, which defeats the purpose. Several approaches have been proposed to tackle this problem. One naive yet common way is to not wait for all machines, and ignore the straggling machines. One may hope that in this way, on average, the

taskmaster receives enough information from everyone; however, in many cases, the overall performance may be negatively impacted because of the lost updates. An alternative and more appropriate way to resolve this issue is to introduce some *redundancy* in the computation of the machines, in order to efficiently trade off computation time for less wait time, and to be able to recover the correct update using only a few machines. Over the past few decades, *coding theory* was developed to address similar challenges in other domains such as mobile communication, storage, data transmission, and broadcast systems. Recently, [198] considered a distributed gradient descent setting for large-scale machine learning, and proposed the idea of *gradient coding*, which uses coding theory to cleverly distribute each gradient iteration across a number of machines in an efficient way. However, the computational complexity of their decoding algorithm was quite high (cubic in the number of returning machines).

In **Chapter 7**, we develop a deterministic scheme that, for a prescribed per-machine computational effort, recovers the gradient from the *least number of machines theoretically permissible*, via a decoding algorithm that is *an order of magnitude faster* than the state of the art. The idea is based on a suitably designed Reed–Solomon code that has a sparsest and balanced generator matrix [88]. Empirical results have demonstrated the clear advantage of our method over competing schemes.

1.5 Synopsis of Part IV: Learning from Data

During the past decade, machine learning, and largely *deep learning*, has made a remarkable impact in many domains, and has enjoyed a great deal of success in a wide variety of tasks, such as computer vision, speech recognition, natural language processing, recommender systems, bioinformatics, and video- and board-game playing. While incredibly successful in many respects, the reasons behind the great success of these methods (as well as their failures in some other respects) are largely unexplained. A *theoretical foundation* that backs these methods is crucial for understanding their capabilities and limitations, and for making them applicable to domains in which they have not been yet successful, and ultimately boosting progress towards intelligent systems.

Due to the nonlinear nature of deep neural networks, their loss function is in general highly *non-convex*. However, empirically, practitioners often obtain zero training error, i.e., a *global minimum* of the training loss, across various datasets, architectures, and settings [224]. This phenomenon is due to the heavy *overparameterization*

present in typical deep models ("tens of millions" of parameters for "tens of thousands" of data points). In other words, these highly overparameterized models have a lot of capacity, which allows them to perfectly fit/interpolate the training data, so much so that this regime has been called the *interpolating* regime [138]. What is surprising, however, is that, contrary to the conventional wisdom which says achieving zero training error (aka overfitting) is harmful for generalization (out-of-sample error), these deep models, trained with simple stochastic gradient descent (SGD) or its variants, *generalize* quite well to unseen data. The loss function of these deep models has in fact (infinitely) many *global minima*, which can have drastically different generalization properties (in fact, there are many global minima that perform very poorly on the test set), and stochastic descent algorithms seem to converge to "special" ones that generalize well, even in the absence of any explicit regularization or early stopping [224].

In an attempt to shed some light on why this is the case, in **Chapter 8**, we shall revisit some minimax properties of stochastic gradient descent (SGD) for the square loss of linear models—originally developed in the 1990s—and extend them to general *stochastic mirror descent (SMD)* algorithms for *general* loss functions and *nonlinear* models. In particular, we show that there is a fundamental identity which holds for SMD (and SGD) under very general conditions, and which implies the minimax optimality of SMD (and SGD) for sufficiently small step size, and for a general class of loss functions and general nonlinear models. We further show that this identity can be used to naturally establish other properties of SMD (and SGD), namely convergence and *implicit regularization* for over-parameterized linear models (in what is now being called the "interpolating regime"), some of which have been shown in certain cases in prior literature.

In **Chapter 9**, we show that, for highly overparameterized *nonlinear* models, the SMD algorithm for any particular potential function converges to a global minimum that is approximately *the closest one to the initialization, in terms of the Bregman divergence corresponding to the potential used.* For the special case of SGD, this means that it converges to a global minimum which is approximately the closest one to the initialization in the usual Euclidean sense. This result further implies that, when initialized around zero, *SGD acts as an ℓ_2-norm regularizer*, a phenomenon referred to as implicit regularization (in linear models [85, 194]). Similarly, by choosing other mirrors, one obtains different forms of implicit regularization, which may have different performances on the test data. Our experimental results

indeed showed a clear difference in the generalization performance of the solutions obtained via different SMD regularizers. Experimenting on the CIFAR-10 dataset with different regularizers, ℓ_1 norm (to encourage sparsity), ℓ_2 norm (SGD, to encourage small Euclidean norm), and ℓ_{10} norm (to discourage large components), consistently showed that the minimum-ℓ_{10}-norm interpolating solution has a better generalization performance than the minimum-ℓ_2-norm one, which in turn has a better generalization performance than the minimum-ℓ_1-norm solution. This surprising result strongly suggests the importance of developing a generalization theory for the overparameterized/interpolating regime and the choice of regularizers.

In **Chapter 10**, we exhibit a new interpretation of SMD, namely that it is a *risk-sensitive optimal* estimator when the unknown weight vector and additive noise are non-Gaussian and belong to the exponential family of distributions. The analysis also suggests a modified version of SMD, which we refer to as symmetric SMD (SSMD). The proofs rely on some simple properties of Bregman divergence, which allow us to extend results from quadratics and Gaussians to certain convex functions and exponential families in a rather seamless way. Furthermore, for vanishing step size SMD, and in the standard stochastic optimization setting, we give a direct and elementary proof of convergence for SMD to the "true" parameter vector, which avoids ergodic averaging or appealing to stochastic differential equations.

Part I

Network Dynamics

Chapter 2

EPIDEMICS OVER COMPLEX NETWORKS: ANALYSIS OF EXACT AND APPROXIMATE MODELS

[1] Navid Azizan and Babak Hassibi. "SIRS epidemics on complex networks: Concurrence of exact Markov chain and approximated models". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015, pp. 2919–2926. DOI: 10.1109/CDC.2015.7402660.

[2] Navid Azizan et al. "Analysis of exact and approximated epidemic models over complex networks". In: *arXiv preprint arXiv:1609.09565* (2016). URL: http://arxiv.org/abs/1609.09565.

Understanding and analyzing the dynamics of networks is crucial for developing societal systems that are robust and efficient. Here, we study one of the most important families of network dynamics, namely, that of spreading processes, or epidemics. We consider the spread of discrete-time epidemics over arbitrary networks for well-known propagation models, namely SIS, SIRS, SEIRS, SIV, SEIV, and their variants. Such spreading processes can be normally described by Markov chains with an exponential number of states in the number of nodes. Since analyzing these Markov chain models is complicated, various linear and nonlinear lower-dimensional approximations of them have been proposed and studied in the literature. The most common of these is the nonlinear "mean-field" approximation and its linearization around the disease-free fixed point, whose numbers of states are linear in the number of nodes. The results we review are for both the exact models and the approximated ones, with a focus on the connections between them. Since the linear model is the first-order approximation of the nonlinear mean-field one at the disease-free equilibrium, its stability determines the local stability of the nonlinear model. Furthermore, for most propagation models, the global dynamics of the nonlinear model also coincides with the stability of the linear model, and takes on one of two forms: either the epidemic dies out, or it converges to another unique fixed point (the so-called endemic state where a constant fraction of the nodes remain infected). We tie in these approximations to the exact Markov chain model. We show that the linear model provides an upper-bound on the true marginal probabilities of infection, and that (in a certain regime) it is the tightest upper-bound that involves only marginals. Furthermore, even though the nonlinear model is not an upper-bound on the true probabilities in general, it does provide an

upper-bound on the probability of the chain not being in the all-healthy state. These bounds also imply the well-known result of sublinear mixing time of the Markov chain (epidemic extinction) when the disease-free fixed point is globally stable in the mean-field model. We compare these results for different propagation models in detail and provide a concise summary of them.

2.1 Introduction

Epidemic models have been extensively studied since a first mathematical formulation was introduced in 1927 by Kermack and McKendrick [113]. Although the classical models mostly neglected the underlying network structure, and assumed a uniformly mixed population, a huge body of work on more realistic networked models has emerged in the recent years. Modeling and analysis of epidemics plays a key role in many areas such as epidemiology [29], information propagation [109, 56], viral marketing [168, 175], and network security [8, 2]. In particular, the models developed in the literature can be used to understand various spreading processes over networks such as the adoption of an idea or fake news in an online social network (Facebook, Twitter, etc.), the consumption of a new product in a market, or the spread of computer viruses over the Internet. Questions of interest include the existence of fixed points, stability (whether the epidemic dies out or spreads), transient behavior, the cost of an epidemic [40, 43], how best to control an epidemic [66, 156], etc.

As in the majority of the literature, we adopt the terminologies of infectious diseases throughout this chapter. The results that we survey are for well-known propagation models, namely SIS (susceptible-infected-susceptible), SIRS (susceptible-infected-recovered-susceptible), SEIRS (susceptible-exposed-infected-recovered-susceptible), SIV (susceptible-infected-vaccinated), and SEIV (susceptible-exposed-infected-vaccinated). In the basic SIS model, each node in the network is in one of two different states: susceptible (healthy) or infected. A healthy node has a chance of getting infected if it has infected neighbors in the network. The probability of getting infected increases as the number of infected neighbors increases. An infected node also has a chance of recovering, after which it still has a chance of getting infected by its neighbors (the flu is an example of this model). SIR and SIRS models have an extra recovered state, which corresponds to the nodes that have recovered from the disease and are not susceptible to it (mumps and pertussis respectively are examples of SIR and SIRS epidemics [102]). In some models, such as SEIRS, there are two different types of unhealthy states: exposed (E) and infected/infectious (I). The nodes in an exposed state have been exposed to the disease but are not infectious

yet. Additionally, in SIV and SEIV models, there is a random vaccination (either permanent or temporary) which permits direct transition from the susceptible state to the recovered (vaccinated) state.

Even the SIS case, which is the simplest of the above models, for a network with n nodes, yields a Markov chain with 2^n states, sometimes called the exact or "stochastic" model. This is a discrete-space model, as there are two possible states of "0" and "1" for healthy and infected. Ostensibly, because analyzing this Markov chain is too complicated, various n-dimensional linear and non-linear approximations have been proposed in the literature. The most common of these is the n-dimensional non-linear mean-field approximation and its corresponding linearization about the disease-free fixed point, which are often referred to as "deterministic" models. These are continuous-space models, that take real numbers between 0 and 1, which can be understood as the marginal probability for being infected (or the infected fraction of the i-th subpopulation).

It is worth noting that all the above models have also been studied in two different settings: continuous-time and discrete-time. In fact, there are two parallel bodies of work in the literature, on continuous-time (e.g., [76, 205, 72, 187, 127, 157, 65]) and discrete-time (e.g., [79, 208, 57, 5, 170, 6]) epidemics. Even though the models are similar in many aspects, depending on the application in hand, it may make more sense to use one or the other. Here, we choose to focus on discrete-time models, but most of the results have counterparts in continuous-time as well.

The results we review are for both the exact and approximated models, with a focus on the connections between them. It is well known that, in many cases, the linear model is an upper-bound on the nonlinear mean-field approximation. It is also known that, depending on the largest eigenvalue of the underlying graph adjacency matrix and the epidemic parameters, the global dynamics of the mean-field approximation takes on one of two forms: either the epidemic dies out (disease-free fixed point) or it converges to another unique fixed point where a constant fraction of the nodes remain infected (endemic state). As for the exact Markov chain model, the linear model provides an upper-bound on the true marginal probabilities of infection, and we show that this is the *tightest* upper-bound using the marginals only, when the infection probabilities are not too high. Furthermore, even though the nonlinear model is *not* an upper-bound on the true probabilities in general, it does provide an upper-bound on the probability of the chain not being absorbed (some nodes being infected). A consequence of these upper-bounds is that when the approximated model is stable to

the disease-free fixed point, the Markov chain has a mixing time of $O(\log n)$, which means the epidemic dies out fast in the true model as well.

In Section 2.2, we review the main epidemic models introduced in the literature. For each one of these spreading processes, we state the exact Markov chain model, the nonlinear mean-field approximation, and the latter's linearization. Section 2.3 concerns the results on the nonlinear model (and its connection to the linear model) for different spreading processes mentioned earlier. We first describe the case where the epidemic dies out, and then the case where the all-healthy fixed point is not stable and there exists a unique nontrivial fixed point which is typically stable. Returning back to the exact Markov chain model, in Section 2.4, we establish the connection between that and the approximated models. In particular, we first state the connection to the linear model (the tightest upper-bound using marginals, and the sufficient condition for fast mixing), and then proceed to the connection to the nonlinear model (upper-bound on the probability of the chain not being absorbed). We also review the extensions to heterogeneous models in Section 2.5, and mention higher-order (such as pairwise) approximations in Section 2.6. We finally summarize and conclude in Section 2.7.

2.2 Models

In this section, we discuss some of the main epidemic models. Additional models can be found in Appendix 2.A.

2.2.1 Susceptible-Infected-Susceptible (SIS)

2.2.1.1 Exact Markov Chain Model

Let $G = (V, E)$ be an arbitrary connected undirected network with n nodes, and with adjacency matrix A. Each node can be in a state of health (S), represented by 0, or a state of infection (I), represented by 1 (see Fig. 2.1). The state of the entire network can be represented by a binary n-tuple $\xi(t) = (\xi_1(t), \cdots, \xi_n(t)) \in \{0, 1\}^n$, where each of the entries represents the state of a node at time t, i.e., i is infected if $\xi_i(t) = 1$, and it is healthy if $\xi_i(t) = 0$.

Given the current state $\xi(t)$, the infection probability of each node in the next step is determined independently, and therefore the transition matrix S of this Markov chain has elements $S_{X,Y} = \mathbb{P}(\xi(t+1) = Y | \xi(t) = X)$ of the following form:

$$\mathbb{P}(\xi(t+1) = Y | \xi(t) = X) = \prod_{i=1}^{n} \mathbb{P}(\xi_i(t+1) = Y_i | \xi(t) = X), \tag{2.1}$$

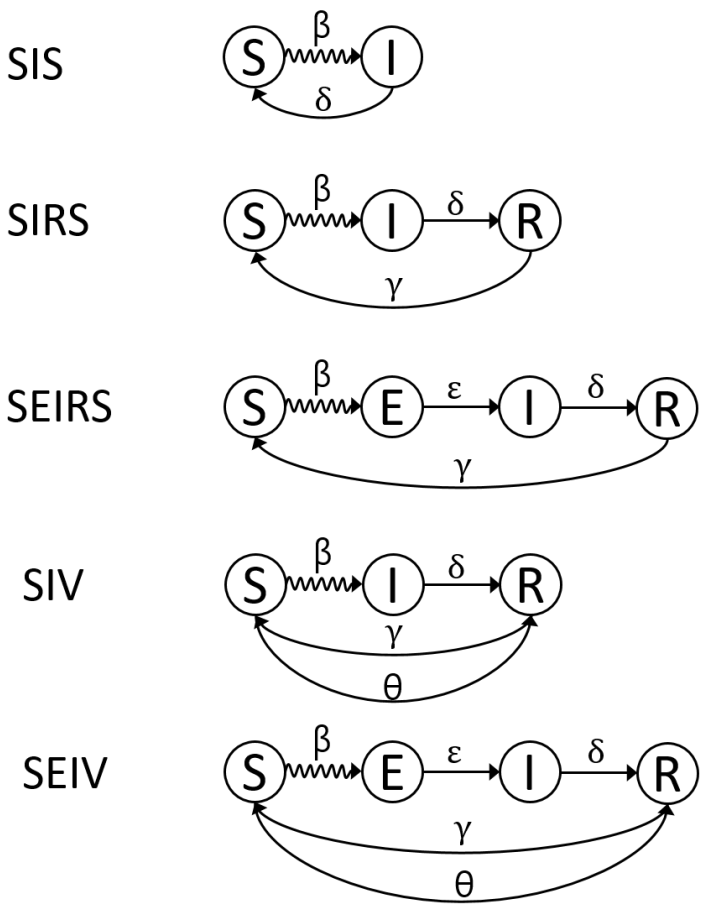


Figure 2.1: State diagram of a single node in different models. Wavy arrows represent exogenous (network-based) transition. S stands for susceptible (healthy), E for exposed, I for infected/infectious, and R for recovered.

for any two state vectors $X, Y \in \{0,1\}^n$.

A healthy node remains healthy if all its neighbors are healthy. During each time epoch, nodes in the healthy (susceptible) state can be infected by their infected neighbors according to independent events with probability β (the *infection rate*) each. Moreover, nodes that are infected can recover during each such time epoch with probability δ (the *recovery rate*), if they do not get infected again at the same time. That is

$$\mathbb{P}(\xi_i(t+1) = Y_i | \xi(t) = X) = \begin{cases} (1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (0,0) \\ 1-(1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (0,1) \\ \delta(1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (1,0) \\ 1-\delta(1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (1,1) \end{cases}, \tag{2.2}$$

where $m_i = |N_i \cap \mathbb{S}(X)|$, where N_i denotes the set of neighbors of node i, and

$\mathbb{S}(X) = \{i : X_i = 1\}$ is the support of $X \in \{0,1\}^n$, i.e., the set of infected nodes.

This Markov chain has a unique absorbing state, which is the state where all the nodes in the network are healthy with probability 1. This is an absorbing state since there is a non-zero probability of reaching it from any other state in a single step, and because once all the nodes are healthy, no node will be exposed to the disease, and they will always stay healthy. This means that the disease will die out if we wait long enough. However, this result is not very revealing, since it may take a long time for the disease to die out, and therefore, the more important question is whether the Markov chain is fast-mixing or whether its mixing time is exponentially large.

Comparing the discrete-time Markov chain model to the continuous-time Markov chain model described in [76], the continuous-time Markov chain model allows only one flip of each node's epidemic state at each moment. However, the discrete-time model allows change of epidemic states for more than one node at each time step. The reason being that the change of epidemic state for two or more nodes can occur at same time interval, even though they do not happen at the same moment. The transition matrix of the embedded Markov chain of the continuous-time model has nonzero entries only where the Hamming distance of the row coordinate and the column coordinate is 1 (they differ in only one element). However, the transition matrix of the discrete-time Markov chain model can have nonzero entries everywhere (except the row of the absorbing state).

Let us denote the probability that node i is infected at time t by $p_i(t) = \mathbb{P}(\xi_i(t) = 1)$. The probability of node i being infected at time $t+1$ can then be written as

$$\begin{aligned} p_i(t+1) = &\mathbb{P}(\xi_i(t+1) = 1 | \xi_i(t) = 1)\mathbb{P}(\xi_i(t) = 1) \\ &+ \mathbb{P}(\xi_i(t+1) = 1 | \xi_i(t) = 0)\mathbb{P}(\xi_i(t) = 0). \end{aligned} \tag{2.3}$$

By marginalizing out the state of the other nodes, we can write this as

$$\begin{aligned} p_i(t+1) = &\mathbb{E}_{\xi_{-i}(t)|\xi_i(t)=1}\left[1 - \delta \prod_{j \in N_i} (1 - \beta \mathbb{1}_{\xi_j(t)=1})\right] p_i(t) \\ &+ \mathbb{E}_{\xi_{-i}(t)|\xi_i(t)=0}\left[1 - \prod_{j \in N_i} (1 - \beta \mathbb{1}_{\xi_j(t)=1})\right] (1 - p_i(t)), \end{aligned} \tag{2.4}$$

where the conditional expectations are on the joint probability of all nodes other than i (denoted by ξ_{-i}).

2.2.1.2 Nonlinear Model

In order to propagate the previous recursion, one needs all the joint probabilities. An approximate model that requires only knowledge of the marginals is the so-called mean-field approximation (MFA):

$$P_i(t+1) = \left(1 - \delta \prod_{j \in N_i} \left(1 - \beta P_j(t)\right)\right) P_i(t) + \left(1 - \prod_{j \in N_i} \left(1 - \beta P_j(t)\right)\right) (1 - P_i(t))$$
$$= (1-\delta)P_i(t) + (1 - (1-\delta)P_i(t)) \left(1 - \prod_{j \in N_i} (1 - \beta P_j(t))\right). \tag{2.5}$$

This approximate model assumes that the events that the neighbors are infected are independent. We use capital P for the approximated probabilities, to distinguish them from the exact probabilities of the Markov chain, p.

It is sometimes convenient to define and work with a map $\Phi : [0,1]^n \to [0,1]^n$ with elements defined as

$$\Phi_i(x) = (1-\delta)x_i + (1 - (1-\delta)x_i) \left(1 - \prod_{j \in N_i} (1 - \beta x_j)\right). \tag{2.6}$$

It is trivial to check that the MFA becomes $P_i(t+1) = \Phi_i([P_1(t), \ldots, P_n(t)]^T)$. The MFA is an n-dimensional model and is therefore computationally much less demanding than the true 2^n-dimensional model. The MFA has been studied in [57, 208, 5] among others. The origin ($P_1(t) = \ldots, = P_n(t) = 0$) is a trivial fixed point of the above model, which is consistent with the absorbing state of the Markov chain model.

2.2.1.3 Linear Model

One step further is to approximate the preceding equations by linearizing (2.5) around the origin, which results in the following mapping:

$$\tilde{P}_i(t+1) = (1-\delta)\tilde{P}_i(t) + \beta \left(\sum_{j \in N_i} \tilde{P}_j(t)\right) \tag{2.7}$$

Putting together the equations of this form for all i, one can see this as

$$\tilde{P}(t+1) = ((1-\delta)I_n + \beta A)\tilde{P}(t), \tag{2.8}$$

where $\tilde{P}(t) = [\tilde{P}_1(t), \ldots, \tilde{P}_n(t)]^T$.

Note that $(1-\delta)I_n + \beta A$ is in fact the Jacobian of the nonlinear model at the origin.

2.2.2 Susceptible-Infected-Recovered-Susceptible (SIRS)

In the SIRS model, there is an additional "recovered" (R) state (see Fig. 2.1). As before, nodes in the susceptible state can be infected by their infected neighbors according to independent events with probability β each. Nodes that are infected can recover with probability δ, and nodes in the recovered state can randomly transition to the susceptible state with probability γ (*immunization loss*).

2.2.2.1 Exact Markov Chain Model

We start again with the exact Markov chain model. The state of node i at time t, denoted by $\xi_i(t)$, can take one of the following values: 0 for *Susceptible*, 1 for *Infected* (or Infectious), and 2 for *Recovered*, i.e., $\xi_i(t) \in \{0, 1, 2\}$.

The state of the entire network can be represented as:

$$\xi(t) = (\xi_i(t), \ldots, \xi_n(t)) \in \{0, 1, 2\}^n . \tag{2.9}$$

The $3^n \times 3^n$ state transition matrix S of the Markov chain has elements of the form

$$S_{X,Y} = \mathbb{P}\left(\xi(t+1) = Y \mid \xi(t) = X\right) = \prod_{i=1}^{n} \mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right), \tag{2.10}$$

where

$$\mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right) = \begin{cases} (1-\beta)^{m_i}, & \text{if } (X_i, Y_i) = (0,0) \\ 1-(1-\beta)^{m_i}, & \text{if } (X_i, Y_i) = (0,1) \\ 0, & \text{if } (X_i, Y_i) = (0,2) \\ 0, & \text{if } (X_i, Y_i) = (1,0) \\ 1-\delta, & \text{if } (X_i, Y_i) = (1,1) \\ \delta, & \text{if } (X_i, Y_i) = (1,2) \\ \gamma, & \text{if } (X_i, Y_i) = (2,0) \\ 0, & \text{if } (X_i, Y_i) = (2,1) \\ 1-\gamma, & \text{if } (X_i, Y_i) = (2,2) \end{cases}, \tag{2.11}$$

where $m_i = \left|\left\{j \in N_i \mid X_j = 1\right\}\right| = |N_i \cap I(t)|$. The set of susceptible, infected, and recovered nodes at time t are denoted by $S(t)$, $I(t)$, and $R(t)$, respectively.

The marginal probabilities can be expressed as $p_{R,i}(t)$ and $p_{I,i}(t)$, for the probability that *node i is in state R at time t* and the probability that *node i is in state I at*

time t, respectively. Then, $p_{S,i}(t)$ is determined by the other two probabilities as $1 - p_{R,i}(t) - p_{I,i}(t)$. Based on the above-mentioned transition rates, we can calculate the two marginal probabilities as

$$p_{R,i}(t+1) = (1-\gamma)p_{R,i}(t) + \delta p_{I,i}(t), \tag{2.12}$$

$$p_{I,i}(t+1) = (1-\delta)p_{I,i}(t) + \mathbb{E}_{|\xi_i(t)=0}\left[1 - \prod_{j\in N_i}(1-\beta\mathbb{1}_{\xi_j(t)=1})\right](1 - p_{R,i}(t) - p_{I,i}(t)). \tag{2.13}$$

The recursion for $p_{S,i}(t+1)$ can be found from $p_{S,i}(t) + p_{I,i}(t) + p_{R,i}(t) = 1$.

2.2.2.2 Nonlinear Model

The mean-field approximation of the above marginal probabilities has been commonly considered, which can be expressed as

$$P_{R,i}(t+1) = (1-\gamma)P_{R,i}(t) + \delta P_{I,i}(t), \tag{2.14}$$

$$P_{I,i}(t+1) = (1-\delta)P_{I,i}(t) + \left(1 - \prod_{j\in N_i}(1-\beta P_{I,j}(t))\right)(1 - P_{R,i}(t) - P_{I,i}(t)). \tag{2.15}$$

This MFA is in fact a nonlinear mapping with $2n$ states (rather than 3^n states).

2.2.2.3 Linear Model

Linearizations of Eqs. (2.14) and (2.15) around the origin can be considered as well, which results in the mapping

$$\tilde{P}_{R,i}(t+1) = (1-\gamma)\tilde{P}_{R,i}(t) + \delta\tilde{P}_{I,i}(t), \tag{2.16}$$

$$\tilde{P}_{I,i}(t+1) = (1-\delta)\tilde{P}_{I,i}(t) + \beta\sum_{j\in N_i}\tilde{P}_{I,j}(t). \tag{2.17}$$

These equations for all i can be expressed in a matrix form as

$$\begin{bmatrix}\tilde{P}_R(t+1)\\ \tilde{P}_I(t+1)\end{bmatrix} = M\begin{bmatrix}\tilde{P}_R(t)\\ \tilde{P}_I(t)\end{bmatrix}, \tag{2.18}$$

where

$$M = \begin{bmatrix}(1-\gamma)I_n & \delta I_n\\ 0_{n\times n} & (1-\delta)I_n + \beta A\end{bmatrix}. \tag{2.19}$$

2.2.3 Susceptible-Infected-Vaccinated (SIV)

2.2.3.1 Exact Markov Chain Model

The SIV model accounts for the effect of vaccination by incorporating direct immunization into the SIRS model. In other words, the transition from S to R is also permitted in this model (see Fig. 2.1). Depending on the value of γ, this model can represent temporary ($\gamma \neq 0$) or permanent ($\gamma = 0$) immunization. Based on the efficacy of the vaccine, there are two variants of this model: infection-dominant and vaccination-dominant.

In the infection-dominant case, if a susceptible node receives both infection and vaccine at the same time, it gets infected. In this case, the elements of the state transition matrix are

$$S_{X,Y} = \mathbb{P}\left(\xi(t+1) = Y \mid \xi(t) = X\right) = \prod_{i=1}^{n} \mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right), \quad (2.20)$$

where

$$\mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right) = \begin{cases} (1-\beta)^{m_i}(1-\theta), & \text{if } (X_i, Y_i) = (0,0) \\ 1-(1-\beta)^{m_i}, & \text{if } (X_i, Y_i) = (0,1) \\ (1-\beta)^{m_i}\theta, & \text{if } (X_i, Y_i) = (0,2) \\ 0, & \text{if } (X_i, Y_i) = (1,0) \\ 1-\delta, & \text{if } (X_i, Y_i) = (1,1) \\ \delta, & \text{if } (X_i, Y_i) = (1,2) \\ \gamma, & \text{if } (X_i, Y_i) = (2,0) \\ 0, & \text{if } (X_i, Y_i) = (2,1) \\ 1-\gamma, & \text{if } (X_i, Y_i) = (2,2) \end{cases}, \quad (2.21)$$

and as before $m_i = \left|\left\{j \in N_i \mid X_j = 1\right\}\right| = |N_i \cap I(t)|$. Eq. (2.21) differs from Eq. (2.11) in the first and third cases, and it reduces to the SIRS model for $\theta = 0$.

The steady-state behavior in the presence of immunization is rather different from the previous models, in which all the nodes became susceptible. In this model, once there is no node in the infected state, the Markov chain reduces to a simpler Markov chain, where the nodes are all decoupled. In fact, from that time on, each node has an independent transition probability between S and R. The stationary distribution of each single node is then $P_S^* = \frac{\gamma}{\gamma+\theta}$ and $P_R^* = \frac{\theta}{\gamma+\theta}$ (Fig. 2.2). This MC converges if

$\gamma\theta \neq 1$, and the stationary distribution of each state X is

$$\pi_X = \prod_{i=1}^{n} \left(\frac{\gamma}{\gamma+\theta}\right)^{\mathbb{I}(X_i=0)} \cdot 0^{\mathbb{I}(X_i=1)} \cdot \left(\frac{\theta}{\gamma+\theta}\right)^{\mathbb{I}(X_i=2)}.$$

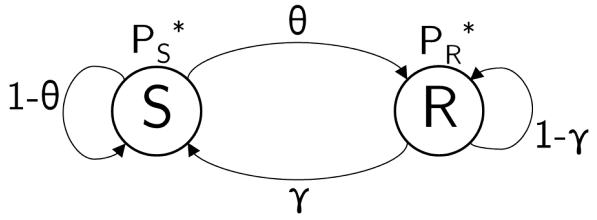


Figure 2.2: Reduced Markov chain of a single node in the steady state.

2.2.3.2 Nonlinear Model

The nonlinear map (mean-field approximation of the Markov chain model) can be obtained as:

$$\begin{aligned} P_{R,i}(t+1) =& (1-\gamma)P_{R,i}(t) + \delta P_{I,i}(t) \\ &+ \prod_{j\in N_i} (1-\beta P_{I,j}(t))\theta(1-P_{R,i}(t)-P_{I,i}(t)), \end{aligned} \tag{2.22}$$

$$\begin{aligned} P_{I,i}(t+1) =& (1-\delta)P_{I,i}(t) + \\ &\left(1-\prod_{j\in N_i}(1-\beta P_{I,j}(t))\right)(1-P_{R,i}(t)-P_{I,i}(t)). \end{aligned} \tag{2.23}$$

It can be easily verified that one fixed point of this nonlinear map occurs at $P_{R,i}(t) = P_R^*$ and $P_{I,i}(t) = 0$, i.e.,

$$\begin{bmatrix} P_R(t) \\ P_I(t) \end{bmatrix} = \begin{bmatrix} \frac{\theta}{\gamma+\theta} 1_n \\ 0_n \end{bmatrix},$$

which is consistent with the steady state of the Markov chain.

2.2.3.3 Linear Model

After some algebra, the linearization of the above model around the fixed point can be expressed as:

$$\begin{bmatrix} \tilde{P}_R(t+1) \\ \tilde{P}_I(t+1) \end{bmatrix} = \begin{bmatrix} P_R^* 1_n \\ 0_n \end{bmatrix} + M \begin{bmatrix} \tilde{P}_R(t) - P_R^* 1_n \\ \tilde{P}_I(t) - 0_n \end{bmatrix}, \tag{2.24}$$

where

$$M = \begin{bmatrix} (1-\gamma-\theta)I_n & (\delta-\theta)I_n - \theta P_S^* \beta A \\ 0_{n\times n} & (1-\delta)I_n + P_S^* \beta A \end{bmatrix}. \tag{2.25}$$

2.3 Results on the Nonlinear MFA Model

The nonlinear mean-field approximation has been extensively studied in the literature for different propagation models. We review the most important results here, starting from the SIS epidemics.

2.3.1 SIS

It is straightforward to see that the linear model upper bounds the nonlinear one, as follows.

$$\begin{aligned} P_i(t+1) =& (1-\delta)P_i(t) + (1-(1-\delta)P_i(t))\left(1 - \prod_{j\in N_i}(1-\beta P_j(t))\right) \\ \leq& (1-\delta)P_i(t) + \left(1 - \prod_{j\in N_i}(1-\beta P_j(t))\right) \\ \leq& (1-\delta)P_i(t) + \beta\left(\sum_{j\in N_i} P_j(t)\right) \end{aligned}$$

For two real-valued column vectors $u, v \in \mathbb{R}^n$, we use the notation $u \preceq v$ to indicate $u_i \leq v_i$ for all $i \in \{1,\ldots,n\}$, and $u \prec v$, if the inequalities are strict. Defining $P(t) = [P_1(t),\ldots,P_n(t)]^T$, we have

$$P(t+1) \preceq ((1-\delta)I_n + \beta A)P(t), \tag{2.26}$$

which leads to the following well-known result.

Proposition 1. *If* $\frac{\beta\lambda_{\max}(A)}{\delta} < 1$*, the origin is a globally asymptotically stable fixed point for both the linear SIS model* (2.8) *and the nonlinear SIS model* (2.5)*.*

The origin, the trivial fixed point of the nonlinear model, is unstable when $\lambda_{max}((1-\delta)I_n + \beta A) > 1$. Moreover, if so, it is not clear in general whether there exists any

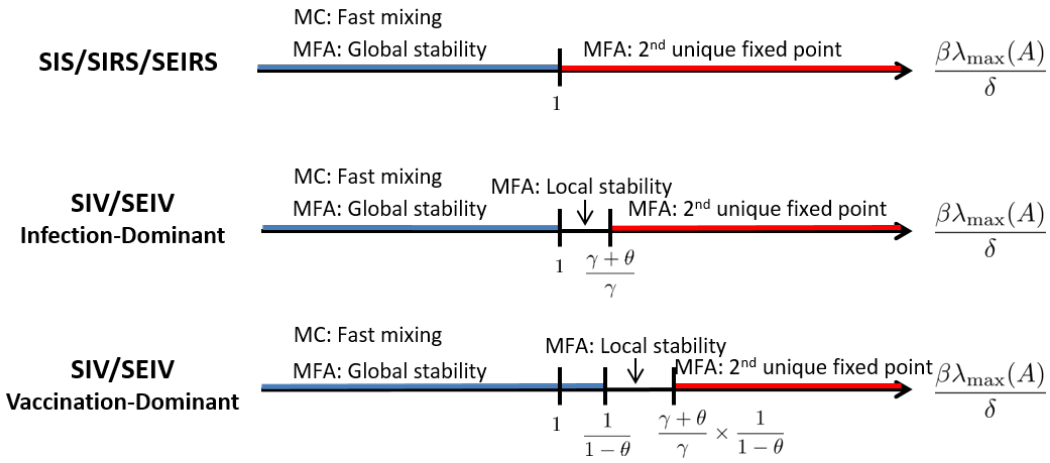


Figure 2.3: Summary of known results for different models. The results have been illustrated as a function of $\frac{\beta\lambda_{max}(A)}{\delta}$. MC stands for the Markov chain model. MFA stands for the mean-field approximation (the nonlinear model).

other fixed point, or how many fixed points there are. It has been shown in the literature (e.g., in [5]) that there exists a unique nontrivial fixed point, and it is stable.

Theorem 2. *If* $\frac{\beta\lambda_{\max}(A)}{\delta} > 1$*, the nonlinear SIS model* (2.5) *has a second unique fixed point. Furthermore, the fixed point is globally asymptotically stable from all initial points (except the origin).*

2.3.2 SIRS/SEIRS/Immune-Admitting-SIS

Similar to the previous (immune-free) SIS model, for the immune-admitting-SIS, SIRS, and SEIRS epidemics, the linear model is an upper-bound on the nonlinear one, and therefore the origin is stable for the nonlinear model when the linear model is stable.

Proposition 3. *If* $\frac{\beta\lambda_{\max}(A)}{\delta} < 1$*, the origin is a globally asymptotically stable fixed point for both the linear model and the nonlinear model for immune-admitting-SIS, SIRS, and SEIRS epidemics.*

In this case, when the origin is not stable, even though there still exists a unique nontrivial fixed point, it is not stable in general [5, 16].

Theorem 4. *If* $\frac{\beta\lambda_{\max}(A)}{\delta} > 1$*, the nonlinear model for immune-admitting-SIS, SIRS, and SEIRS epidemics has a second unique fixed point.*

The following is an example of an unstable nontrivial fixed point for the immune-admitting-SIS model [4, p. 64].

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad \beta = 0.9 \quad \delta = 0.9 \tag{2.27}$$

The nontrivial fixed point of the system above is $P^* = (0.286, 0.222, 0.222)^T$. The linearized model around P^* is

$$\begin{pmatrix} -0.260 & 0.514 & 0.514 \\ 0.700 & -0.157 & 0 \\ 0.700 & 0 & -0.157 \end{pmatrix},$$

which has an eigenvalue of -1.059, which means that P^* is not locally stable. It can be shown that for any initial condition other the origin and P^*, $P(t)$ converges to a cycle.

Even though the nontrivial fixed point is not stable in general, it is known to be stable with high probability for a general family of random graphs [4, p. 66].

Theorem 5 (Ahn and Hassibi [5])**.** *Suppose that $G^{(n)}$ is a random graph with n vertices, and let $d_{\min}^{(n)}$ and $d_{\max}^{(n)}$ denote the minimum and maximum degree of $G^{(n)}$. If, for any fixed $a > 0$, $\mathbb{P}((d_{\min}^{(n)})^2 > a \cdot d_{\max}^{(n)})$ goes to 1 as n goes to infinity, then, with high probability, the origin is unstable, and the second fixed point is locally stable, for any fixed β and δ.*

One can think of several random graph models that satisfy the condition of Theorem 5. For example, if the random graph has a uniform degree that grows with n, then the minimum degree and maximum degree are identical and the ratio $\frac{d_{\min}^2}{d_{\max}} = d$ will grow with any n and exceed a with high probability. Similarly, for random graphs where the degree distribution of the nodes are identical and concentrate, so that we can expect the maximum degree and the minimum degree to be proportional to the expected degree, $\frac{d_{\min}^2}{d_{\max}}$ grows if the expected degree increases unbounded with n. In particular, the Erdös-Rényi random graph $G^{(n)} = G(n, p(n))$ has identical degree distribution, and we have the following result [4, p. 69].

Corollary 6 (Ahn and Hassibi [5])**.** *Consider an Erdös-Rényi random graph $G^{(n)} = G(n, p(n))$ with $p(n) = c\frac{\log n}{n}$ where $c > 1$ is a constant. The nonlinear model is*

locally unstable at the origin and has a locally stable nontrivial fixed point with high probability for any fixed β and δ.

Since $p = c\frac{\log n}{n}$ for $c = 1$ is also the threshold for connectivity, we can say that connected Erdös-Rényi graphs have a nontrivial stable fixed point with high probability.

The random geometric graph $G^{(n)} = G(n, r(n))$ also has identical degree distribution if each node is distributed uniformly. Such random graphs have maximum and minimum degrees which are proportional to the expected degree with high probability if $r(n)$ is smaller than the threshold of connectivity [167], and, similar to Erdös-Rényi graphs, it has high probability of having a nontrivial stable fixed point if the degree grows with n.

2.3.3 SIV/SEIV (Infection-Dominant)

Since the linear model is always the Jacobian of the nonlinear one, its stability determines the local stability of the nonlinear model. However, global stability is harder to show. The following result summarizes the stability of the nonlinear model.

Proposition 7. *The disease-free fixed point of the nonlinear model for the infection-dominant SIV and the infection-dominant SEIV epidemics is*

a) locally stable, if $\frac{\gamma}{\gamma+\theta}\frac{\beta}{\delta}\lambda_{max}(A) < 1$, and

b) globally stable, if $\frac{\beta}{\delta}\lambda_{max}(A) < 1$.

When $\frac{\gamma}{\gamma+\theta}\frac{\beta\lambda_{\max}(A)}{\delta} > 1$, the main fixed point of the nonlinear map is not stable, and again, there exists a unique non-trivial fixed point. This result has been proven in [16] for the SIV case, and it extends to the SEIV model in a similar fashion.

Theorem 8. *If $\frac{\gamma}{\gamma+\theta}\frac{\beta\lambda_{\max}(A)}{\delta} > 1$, the nonlinear model for the infection-dominant SIV and the infection-dominant SEIV epidemics has a second unique fixed point.*

The middle panel in Figure 2.3 shows a summary of the above results.

2.3.4 SIV/SEIV (Vaccination-Dominant)

It is natural to expect the vaccination-dominant models to be more stable than the infection-dominant ones. It turns out that these models are indeed more stable by a factor of $1/(1-\theta)$.

Proposition 9. *The disease-free fixed point of the nonlinear model for the vaccination-dominant SIV and the vaccination-dominant SEIV epidemics is*

a) locally stable, if $(1-\theta)\frac{\gamma}{\gamma+\theta}\frac{\beta}{\delta}\lambda_{max}(A) < 1$, *and*

b) globally stable, if $(1-\theta)\frac{\beta}{\delta}\lambda_{max}(A) < 1$.

Theorem 10. *If* $(1-\theta)\frac{\gamma}{\gamma+\theta}\frac{\beta}{\delta}\lambda_{max}(A) > 1$, *the nonlinear model for the vaccination-dominant SIV and the vaccination-dominant SEIV epidemics has a second unique fixed point.*

Both of the above results have been proven in [16] for the SIV case, and the proof extends to the SEIV case with some modification. The lower panel in Figure 2.3 shows a summary of these results.

2.4 Results on the Exact Markov Chain Model

Returning back to the Markov chain model, in this section, we study the true marginal probabilities of infection and how they relate to the nonlinear and linear models. For the sake of simplicity, we state the results for the SIS model, but they have counterparts for other propagation models as well.

2.4.1 Connection to the Linear Model

It is well-known that the linear model provides an upper-bound on the true marginal probabilities of infection [6].

Proposition 11. *For SIS epidemics, the linear model is an upper-bound on the true marginal probabilities of infection, i.e.,*

$$p_i(t+1) \leq (1-\delta)p_i(t) + \beta \sum_{j \in N_i} p_j(t),$$

for $i = 1, \ldots, n$, *and any time* t.

This can be equivalently expressed in vector form as

$$p(t+1) \preceq ((1-\delta)I_n + \beta A)p(t). \tag{2.28}$$

$(1-\delta)I_n + \beta A$ is the system matrix of the linear model.

A natural question to ask about the above bound is how tight is it. It is clear that the bound cannot be tight in general, as we are ignoring the higher-order dependencies

(e.g., pairwise infection probabilities, etc.). However, we can ask how tight is it among all the bounds that involve only the marginal probabilities. It turns out that when the infection probabilities are not too high, this bound is indeed the tightest bound using marginals only. In other words, if in the SIS model, we maximize $p_i(t+1)$ over all distributions with fixed marginals $p_1(t), \ldots, p_n(t)$, in the low-infection regime, we obtain the same bound.

Theorem 12. *For SIS epidemics, if* $\sum_{i=1}^{n} p_i(t) < 1$*, then the tightest upper-bound on* $p_i(t+1)$ *that involves only the marginal probabilities at time* t *is*

$$p_i(t+1) \leq (1-\delta)p_i(t) + \beta \sum_{j \in N_i} p_j(t),$$

for any $i = 1, \ldots, n$.

It has been shown that when the linear and nonlinear models are stable (i.e., $\frac{\beta\lambda_{\max}(A)}{\delta} < 1$), then the epidemic dies out quickly. More specifically under this condition, the Markov chain has fast "mixing" to the all-healthy state. The mixing time of a Markov chain is defined [126, Def. 4.5] as

$$t_{mix}(\epsilon) = \min\{t : \sup_{\mu} \|\mu S^t - \pi\|_{TV} \leq \epsilon\}, \tag{2.29}$$

where μ is any initial probability distribution defined on the state space, and π is the stationary distribution. $\|\cdot\|_{TV}$ is total variation distance which measures distance of two probability distributions. Total variation distance of two probability measures μ and μ' is defined by

$$\|\mu - \mu'\|_{TV} = \frac{1}{2} \sum_{x} |\mu(x) - \mu'(x)| \tag{2.30}$$

where x is any possible state in the probability space. In fact, $t_{mix}(\epsilon)$ is the smallest time where distance between the stationary distribution and probability distribution at time t from any initial distribution is smaller than or equal to ϵ. Roughly speaking, the mixing time measures how fast initial distribution converges to the limit distribution.

The fast (logarithmic) mixing-time result was first shown by Ganesh et al [76].

Theorem 13. *If* $\frac{\beta\lambda_{\max}(A)}{\delta} < 1$*, the mixing time of the Markov chain whose transition matrix* S *is described by Eqs.* (2.1) *and* (2.2) *is* $O(\log n)$.

The above condition for fast extinction of the epidemic seems to be quite tight in many cases, such as in Erdős-Rényi random graphs. In particular, Figure 2.4 shows

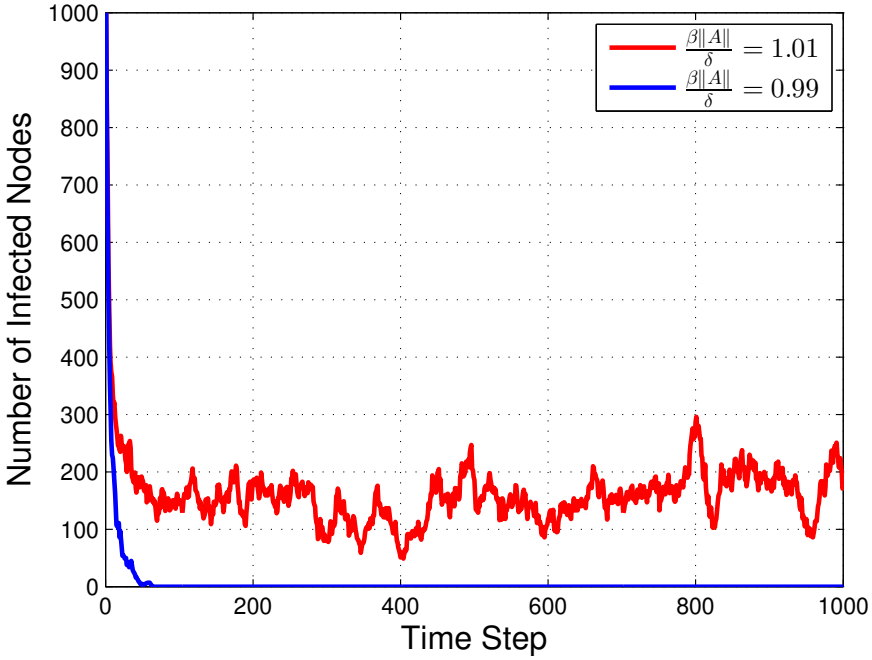


Figure 2.4: A typical example of the evolution of an SIS epidemic over an Erdős-Rényi graph with $n = 2000$ nodes and $\lambda_{\max}(A) = 16.159$. When the condition $\frac{\beta\lambda_{\max}(A)}{\delta} < 1$ is satisfied (e.g., $\beta = 0.055$, $\delta = 0.9$) the epidemic decays exponentially, and dies out quickly (blue curve). In contrast, when $\frac{\beta\lambda_{\max}(A)}{\delta} > 1$ (e.g., $\beta = 0.056$, $\delta = 0.9$), the epidemic does not exhibit convergence to the disease-free state in any observable time (red curve). In fact, the epidemic keeps spreading around the nontrivial fixed point.

a typical simulation of the epidemic in two different cases: (1) When $\frac{\beta\lambda_{\max}(A)}{\delta} = 0.99$ and (2) when $\frac{\beta\lambda_{\max}(A)}{\delta} = 1.01$. As one can see, there is a sharp phase transition happening around this critical value. In other words, the epidemic does not seem to die out in any reasonable amount of time once $\frac{\beta\lambda_{\max}(A)}{\delta}$ is somewhat above 1.

2.4.2 Connection to the Nonlinear Model

The nonlinear model is not an upper-bound on the true probabilities $p_i(t)$ in general. However, it turns out that it does provide an upper-bound on the probability that the chain is not in the all-healthy state (i.e., existence of infection) [4], if one initializes the nonlinear model from the all-infected state.

Theorem 14. *For any time t and any initial state X, we have*

$$\mathbb{P}(\xi(t) \neq \bar{0} | \xi(0) = X) \leq 1 - \prod_{i \in \mathbb{S}(X)} \left(1 - \Phi_i^t(1_n)\right),$$

where $\Phi(.)$ *is the nonlinear approximate model.*

Using this bound, the same mixing time result as in (13) can also be established.

We should finally remark that the reason why it is possible for the nonlinear map to converge to a unique non-origin fixed point when $\frac{\beta\lambda_{\max}(A)}{\delta} > 1$, even though the original Markov chain model always converges to the all-healthy state, is that this is only an upper bound on $\mathbb{P}(\xi(t) \neq \bar{0}|\xi(0) = X)$. In other words, if the origin is globally stable in the epidemic map Φ, we can infer that the Markov chain model mixes fast. However, if the origin in the epidemic map is unstable, we cannot infer anything about the mixing time.

2.5 Heterogeneous Network Models

In the models discussed throughout the chapter, the epidemic parameters $\beta, \delta, \gamma, \epsilon, \theta$ were homogeneous across the network, i.e., they were the same for all nodes. However, many of the results have been extended to more general heterogeneous models.

For the SIS model, much of the behavior of the models was determined by the largest eigenvalue of $M = (1 - \delta)I_n + \beta A$. M is defined by β, the infection rate, δ, the recovery rate, and A, the adjacency matrix. In other words, M is the contact model. To model an epidemic spread where each node has its own infection and recovery rate, we can define a generalized infection matrix. Let $M = (m_{i,j})$ be the generalized infection matrix where $m_{i,j} \in [0, 1]$ represents the infection probability that i is infected at time $t + 1$ when j is the only infected node at time t. In this setting, each diagonal entry $m_{i,i}$ represents self-infection rate. In other words, $1 - m_{i,i}$ is the recovery rate of node i, and $m_{i,i}$ is the probability that i stays infected when there are no other infected nodes in the network. We also assume that probability of infection of each node given the current state $\xi(t)$ is independent. More precisely, for any two state vectors $X, Y \in \{0, 1\}^n$,

$$\mathbb{P}(\xi(t + 1) = Y|\xi(t) = X) = \prod_{i=1}^{n} \mathbb{P}(\xi_i(t + 1) = Y_i|\xi(t) = X) \tag{2.31}$$

Probability transition from given state is defined by M.

$$\mathbb{P}(\xi_i(t + 1) = Y_i|\xi(t) = X) = \begin{cases} \prod\limits_{j\in\mathbb{S}(X)} (1 - m_{i,j}) & \text{if } Y_i = 0, \\ 1 - \prod\limits_{j\in\mathbb{S}(X)} (1 - m_{i,j}) & \text{if } Y_i = 1, \end{cases} \tag{2.32}$$

We define the transition matrix, $S^{(M)} \in \mathbb{R}^{\{0,1\}^n\times\{0,1\}^n}$ by $S^{(M)}_{X,Y} = \mathbb{P}(\xi_i(t + 1) = Y_i|\xi(t) = X)$ in the equation above.

The nonlinear map associated with M, $\Phi^{(M)} : [0,1]^n \to [0,1]^n$ is defined by

$$\Phi_i^{(M)}(x) = 1 - \prod_{j=1}^{n}(1 - m_{i,j}x_j) \tag{2.33}$$

and $\Phi^{(M)} = (\Phi_1^{(M)}, \Phi_2^{(M)}, \cdots, \Phi_n^{(M)})$. M is the Jacobian matrix of $\Phi^{(M)}(\cdot)$ at the origin which gives an upper bound, i.e., $\Phi^{(M)}(x) \preceq Mx$. The origin is the unique fixed point which is globally stable if the largest eigenvalue of M is smaller than 1. It also has a unique nontrivial fixed point which is globally stable if the largest eigenvalue of M is greater than 1.

Similar to the previous cases, $\lambda_{\max}(M) < 1$ guarantees that the mixing time of the Markov chain defined by the transition matrix $S^{(M)}$ is $O(\log n)$.

2.6 Pairwise and Higher-Order Approximate Models

Even though the threshold condition of Theorem 13 for fast extinction of the epidemic (sublinear mixing time of the Markov chain) appears to be tight for the Erdős-Rényi random graph and some other networks, it is known to not be tight for several other cases, such as the star graph [23].

In order to obtain tighter threshold conditions, one can keep track of terms that are of a higher order than the marginals, such as pairwise probabilities of infection, triples, etc. Of course, this comes at the cost of an increased number of states (quadratic, cubic, etc. in the number of nodes), and there is a trade-off between the tightness of the bound and the complexity of the model. In theory, if one takes into account all marginals, pairs, triples, and higher-order terms, we get back to the original exponential-state Markov chain model.

Tighter bounds using pairwise probabilities will be the subject of the next chapter. These bounds were first introduced in [23], and have been extended to heterogeneous models in [160]. Other pairwise approximations have also been studied in [54] but they do not provide any bound on the exact probabilities.

2.7 Summary and Conclusion

We studied the networked SIS, SIRS, SEIRS, SIV, and SEIV epidemics and their variants, using their exact Markov chain models, and their well-known linear and nonlinear mean-field approximations. Below a threshold, the disease-free fixed point is globally stable for the nonlinear model, and also the mixing time of the exact Markov chain is $O(\log n)$, which means the epidemic dies out fast. Furthermore,

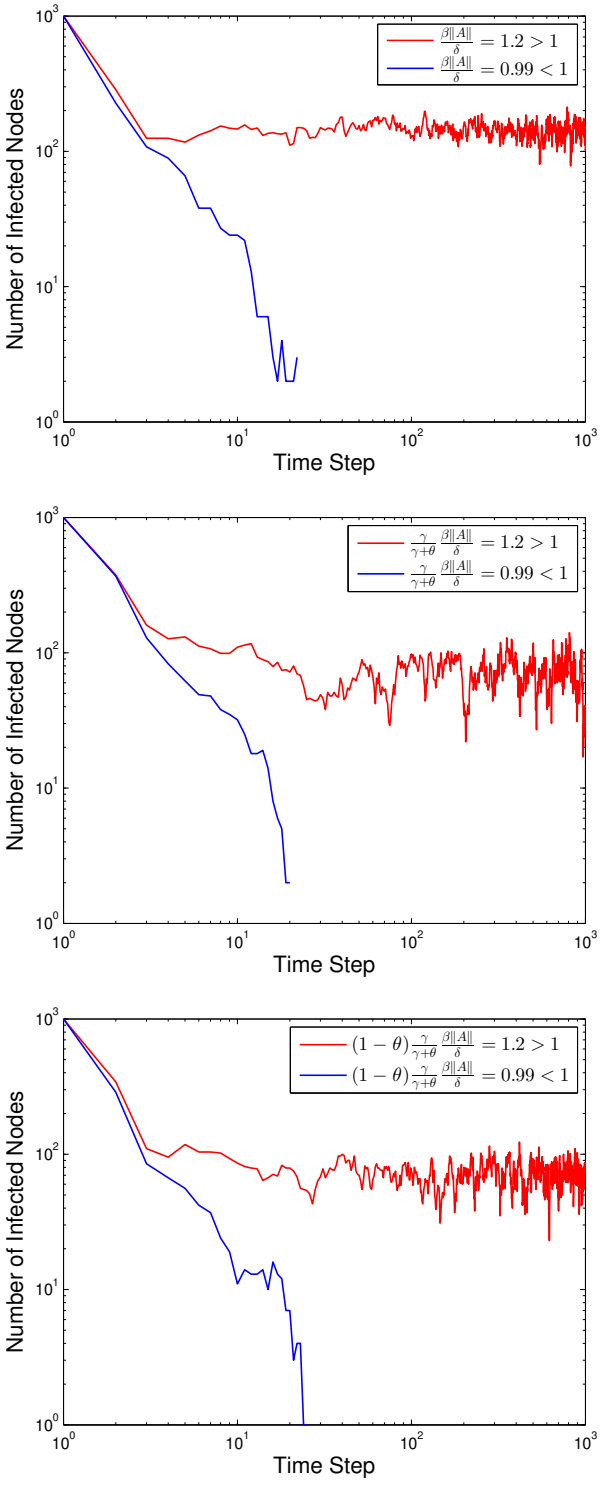


Figure 2.5: The evolution of (a) SIS/SIRS/SEIRS, (b) SIV/SEIV (infection-dominant), (c) SIV/SIEV (vaccination-dominant) epidemics over an Erdős-Rényi graph with $n = 2000$ nodes. The blue curves show fast extinction of the epidemic. The red curves show epidemic spread around the nontrivial fixed point.

above a threshold, the disease-free fixed point is not stable for the linear and nonlinear models, and there exists a second unique fixed point, which corresponds to the endemic state. This nontrivial fixed point is also stable in most cases. Figure 2.3 summarizes all the results.

Typical examples of the spread of the epidemic for all different propagation models studied throughout the chapter have been demonstrated in Figure 2.5. For the SIRS and SEIRS models, the threshold condition is $\frac{\beta\|A\|}{\delta} < 1$, which is the same as that of the SIS one, and it means having an additional recovered state does not necessarily make the system more stable. For the infection-dominant SIV and SEIV models, we observe the same exponential decay of the infection when $\frac{\gamma}{\gamma+\theta}\frac{\beta\|A\|}{\delta} < 1$ (e.g., when $\|A\| = 16.232$ and $\beta = 0.11$, $\gamma = 0.5$ and $\theta = 0.5$), which means the vaccination indeed makes the system more stable. Furthermore, for the vaccination-dominant models, under $(1-\theta)\frac{\gamma}{\gamma+\theta}\frac{\beta\|A\|}{\delta} < 1$ (e.g., $\beta = 0.22$), we observe the fast convergence again, which confirms that the system is even more stable when vaccination is dominant. As plots show, for above-the-threshold cases (e.g., $\beta = 0.07$ for SIRS, 0.13 for SIV-infection-dominant, and 0.29 for SIV-vaccination-dominant), we do not observe epidemic extinction in any reasonable time, and effectively, the epidemic remains endemic.

Finally, we should remark that characterizing the exact epidemic threshold of the Markov chain model is still an open problem. Extensive numerical simulations suggest the existence of such a threshold and a phase transition behavior. Even though in certain networks, such as the Erdős-Rényi random graphs, the epidemic threshold seems to coincide with the condition for local stability of the nonlinear mean-field model (global stability of the linear model), it is different from that condition in general. For this reason, pairwise (and even higher-order) approximations may be sought, which provide tighter bounds on the epidemic threshold.

2.A Additional Models

We review additional epidemic models, namely the immune-admitting variant of the SIS model, the SIERS model, the vaccination-dominant variant of the SIV model, and the SEIV model.

2.A.1 Immune-Admitting SIS

2.A.1.1 Exact Markov Chain Model

A variant of the SIS model is the "immune-admitting" SIS, which is similar to the previous model except that a node does not get infected from its neighbors if it has just recovered from the disease (see Fig. 2.1). In other words, the probability of recovering from the disease is δ. That is

$$\begin{aligned}&\mathbb{P}(\xi_i(t+1) = Y_i | \xi(t) = X)\\ &= \begin{cases} (1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (0,0),\ |N_i \cap \mathbb{S}(X)| = m_i, \\ 1-(1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (0,1),\ |N_i \cap \mathbb{S}(X)| = m_i, \\ \delta & \text{if } (X_i, Y_i) = (1,0), \\ 1-\delta & \text{if } (X_i, Y_i) = (1,1). \end{cases}\end{aligned} \tag{2.34}$$

and, as before, the elements of the transition matrix are defined as

$$\mathbb{P}(\xi(t+1) = Y | \xi(t) = X) = \prod_{i=1}^{n} \mathbb{P}(\xi_i(t+1) = Y_i | \xi(t) = X). \tag{2.35}$$

In this model, the probability that a node becomes healthy from infected (δ) is larger than that of the "immune-free" model ($\delta(1-\beta)^{m_i}$). Therefore, roughly speaking, the immune-admitting model is more likely than the immune-free model to hit the absorbing state.

2.A.1.2 Nonlinear Model

A mean-field approximation for the immune-admitting model can be studied as well, which is defined as

$$P_i(t+1) = (1-\delta)P_i(t) + (1-P_i(t))\left(1 - \prod_{j \in N_i} (1 - \beta P_j(t))\right). \tag{2.36}$$

2.A.1.3 Linear Model

The nonlinear model has the same Jacobian matrix as that of the previous section, which is

$$\tilde{P}(t+1) = ((1-\delta)I_n + \beta A)\tilde{P}(t). \tag{2.37}$$

2.A.2 Susceptible-Exposed-Infected-Recovered-Susceptible (SEIRS)

2.A.2.1 Exact Markov Chain Model

This model has an extra "exposed" state. The state of the nodes can take one of the following values: 0 for *Susceptible*, 1 for *Exposed*, 2 for *Infected* (or Infectious), and 3 for *Recovered* (see Fig. 2.1). The $4^n \times 4^n$ state transition matrix S of the Markov chain has elements of the form

$$S_{X,Y} = \mathbb{P}\left(\xi(t+1) = Y \mid \xi(t) = X\right) = \prod_{i=1}^{n} \mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right), \tag{2.38}$$

as before. Here we have

$$\mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right) = \begin{cases} (1-\beta)^{m_i}, & \text{if } (X_i, Y_i) = (0,0) \\ 1-(1-\beta)^{m_i}, & \text{if } (X_i, Y_i) = (0,1) \\ 1-\epsilon, & \text{if } (X_i, Y_i) = (1,1) \\ \epsilon, & \text{if } (X_i, Y_i) = (1,2) \\ 1-\delta, & \text{if } (X_i, Y_i) = (2,2) \\ \delta, & \text{if } (X_i, Y_i) = (2,3) \\ \gamma, & \text{if } (X_i, Y_i) = (3,0) \\ 1-\gamma, & \text{if } (X_i, Y_i) = (3,3) \\ 0, & \text{otherwise} \end{cases}, \tag{2.39}$$

where $m_i = |N_i \cap I(t)|$.

2.A.2.2 Nonlinear Model

The nonlinear mean-field approximation is

$$\begin{aligned} P_{E,i}(t+1) =& (1-\epsilon)P_{E,i}(t) + \\ & \left(1 - \prod_{j \in N_i} (1 - \beta P_{I,j}(t))\right)(1 - P_{E,i}(t) - P_{I,i}(t) - P_{R,i}(t)) \\ P_{I,i}(t+1) =& \epsilon P_{E,i}(t) + (1-\delta)P_{I,i}(t) \\ P_{R,i}(t+1) =& (1-\gamma)P_{R,i}(t) + \delta P_{I,i}(t). \end{aligned}$$

2.A.2.3 Linear Model

The linearization of the above equations around the origin is

$$\begin{bmatrix} \tilde{P}_E(t+1) \\ \tilde{P}_I(t+1) \\ \tilde{P}_R(t+1) \end{bmatrix} = M \begin{bmatrix} \tilde{P}_E(t) \\ \tilde{P}_I(t) \\ \tilde{P}_R(t) \end{bmatrix},$$

where

$$M = \begin{bmatrix} (1-\epsilon)I_n & \beta A & 0_{n\times n} \\ \epsilon I_n & (1-\delta)I_n & 0_{n\times n} \\ -\theta I_n & \delta I_n & (1-\gamma)I_n \end{bmatrix}.$$

2.A.3 Vaccination-Dominant SIV

2.A.3.1 Exact Markov Chain Model

In the vaccination-dominant variant of the model, the assumption is that if a susceptible node receives both infection and vaccine at the same time, it becomes vaccinated. Although in the context of contagious diseases, this variation might make less sense, in other applications, there are scenarios for which this model is more relevant. The transition probabilities of the Markov chain are again

$$S_{X,Y} = \mathbb{P}\left(\xi(t+1) = Y \mid \xi(t) = X\right) = \prod_{i=1}^{n} \mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right), \quad (2.40)$$

with the change that

$$\mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right) = \begin{cases} (1-\beta)^{m_i}(1-\theta), & \text{if } (X_i, Y_i) = (0,0) \\ (1-(1-\beta)^{m_i})(1-\theta), & \text{if } (X_i, Y_i) = (0,1) \\ \theta, & \text{if } (X_i, Y_i) = (0,2) \\ 0, & \text{if } (X_i, Y_i) = (1,0) \\ 1-\delta, & \text{if } (X_i, Y_i) = (1,1) \\ \delta, & \text{if } (X_i, Y_i) = (1,2) \\ \gamma, & \text{if } (X_i, Y_i) = (2,0) \\ 0, & \text{if } (X_i, Y_i) = (2,1) \\ 1-\gamma, & \text{if } (X_i, Y_i) = (2,2) \end{cases}, \quad (2.41)$$

where $m_i = \left|\left\{j \in N_i \mid X_j = 1\right\}\right| = |N_i \cap I(t)|$, as before.

2.A.3.2 Nonlinear Model

The nonlinear map, or the mean-field approximation, can be stated as:

$$\begin{aligned} P_{R,i}(t+1) =& (1-\gamma)P_{R,i}(t) + \delta P_{I,i}(t) \\ &+ \theta(1 - P_{R,i}(t) - P_{I,i}(t)), \qquad (2.42) \\ P_{I,i}(t+1) =& (1-\delta)P_{I,i}(t) + (1-\theta) \\ &\cdot \Big(1 - \prod_{j\in N_i}(1-\beta P_{I,j}(t))\Big)(1 - P_{R,i}(t) - P_{I,i}(t)). \qquad (2.43) \end{aligned}$$

2.A.3.3 Linear Model

As a result, the first order (linear) model is:

$$\begin{bmatrix} \tilde{P}_R(t+1) \\ \tilde{P}_I(t+1) \end{bmatrix} = \begin{bmatrix} P_R^* 1_n \\ 0_n \end{bmatrix} + M \begin{bmatrix} \tilde{P}_R(t) - P_R^* 1_n \\ \tilde{P}_I(t) - 0_n \end{bmatrix},$$

where

$$M = \begin{bmatrix} (1-\gamma-\theta)I_n & (\delta-\theta)I_n - \theta P_S^* \beta A \\ 0_{n\times n} & (1-\delta)I_n + (1-\theta)P_S^* \beta A \end{bmatrix}.$$

We should note that for the vaccination-dominant model, the steady state of the Markov chain and the main fixed point of the mapping are exactly the same as those of the infection-dominant model. However, as one may expect, it turns out that the vaccination-dominant model is more stable.

2.A.4 SEIV (Infection-Dominant)

The Markov chain model in this case has the following transition probabilities (see Fig. 2.1).

2.A.4.1 Exact Markov Chain Model

$$\mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right) = \begin{cases} (1-\beta)^{m_i}(1-\theta), & \text{if } (X_i, Y_i) = (0,0) \\ 1-(1-\beta)^{m_i}, & \text{if } (X_i, Y_i) = (0,1) \\ (1-\beta)^{m_i}\theta, & \text{if } (X_i, Y_i) = (0,3) \\ 1-\epsilon, & \text{if } (X_i, Y_i) = (1,1) \\ \epsilon, & \text{if } (X_i, Y_i) = (1,2) \\ 1-\delta, & \text{if } (X_i, Y_i) = (2,2) \\ \delta, & \text{if } (X_i, Y_i) = (2,3) \\ \gamma, & \text{if } (X_i, Y_i) = (3,0) \\ 1-\gamma, & \text{if } (X_i, Y_i) = (3,3) \\ 0, & \text{otherwise} \end{cases}, \tag{2.44}$$

where $m_i = |N_i \cap I(t)|$.

2.A.4.2 Nonlinear Model

The nonlinear approximation in this case is

$$\begin{aligned} P_{E,i}(t+1) =& (1-\epsilon)P_{E,i}(t) + \\ & \Big(1 - \prod_{j \in N_i}(1-\beta P_{I,j}(t))\Big)(1 - P_{E,i}(t) - P_{I,i}(t) - P_{R,i}(t)) \\ P_{I,i}(t+1) =& \epsilon P_{E,i}(t) + (1-\delta)P_{I,i}(t) \\ P_{R,i}(t+1) =& (1-\gamma)P_{R,i}(t) + \delta P_{I,i}(t) \\ & + \Big(\prod_{j \in N_i}(1-\beta P_{I,j}(t))\Big)\theta(1 - P_{E,i}(t) - P_{I,i}(t) - P_{R,i}(t)). \end{aligned}$$

2.A.4.3 Linear Model

The linearization around the main fixed point is as follows

$$\begin{bmatrix} \tilde{P}_E(t+1) \\ \tilde{P}_I(t+1) \\ \tilde{P}_R(t+1) \end{bmatrix} = \begin{bmatrix} 0_n \\ 0_n \\ P_R^* 1_n \end{bmatrix} + M \begin{bmatrix} \tilde{P}_E(t) \\ \tilde{P}_I(t) \\ \tilde{P}_R(t) - P_R^* 1_n \end{bmatrix},$$

where

$$M = \begin{bmatrix} (1-\epsilon)I_n & P_S^*\beta A & 0_{n\times n} \\ \epsilon I_n & (1-\delta)I_n & 0_{n\times n} \\ -\theta I_n & (\delta-\theta)I_n - \theta P_S^*\beta A & (1-\gamma-\theta)I_n \end{bmatrix}.$$

2.A.5 SEIV (Vaccination-Dominant)

2.A.5.1 Exact Markov Chain Model

The vaccination-dominant variant of the model has the following transition probabilities.

$$\mathbb{P}\left(\xi_i(t+1) = Y_i \mid \xi(t) = X\right) = \begin{cases} (1-\beta)^{m_i}(1-\theta), & \text{if } (X_i, Y_i) = (0,0) \\ (1-(1-\beta)^{m_i})(1-\theta), & \text{if } (X_i, Y_i) = (0,1) \\ \theta, & \text{if } (X_i, Y_i) = (0,3) \\ 1-\epsilon, & \text{if } (X_i, Y_i) = (1,1) \\ \epsilon, & \text{if } (X_i, Y_i) = (1,2) \\ 1-\delta, & \text{if } (X_i, Y_i) = (2,2) \\ \delta, & \text{if } (X_i, Y_i) = (2,3) \\ \gamma, & \text{if } (X_i, Y_i) = (3,0) \\ 1-\gamma, & \text{if } (X_i, Y_i) = (3,3) \\ 0, & \text{otherwise} \end{cases}, \tag{2.45}$$

where $m_i = |N_i \cap I(t)|$.

2.A.5.2 Nonlinear Model

The nonlinear mean-field approximation can be expressed as

$$\begin{aligned} &P_{E,i}(t+1) = (1-\epsilon)P_{E,i}(t) + (1-\theta)\times \\ &\left(1 - \prod_{j \in N_i}(1-\beta P_{I,j}(t))\right)(1 - P_{E,i}(t) - P_{I,i}(t) - P_{R,i}(t)) \\ &P_{I,i}(t+1) = \epsilon P_{E,i}(t) + (1-\delta)P_{I,i}(t) \\ &P_{R,i}(t+1) = (1-\gamma)P_{R,i}(t) + \delta P_{I,i}(t) \\ &+ \theta(1 - P_{E,i}(t) - P_{I,i}(t) - P_{R,i}(t)). \end{aligned}$$

2.A.5.3 Linear Model

The linearized model is

$$\begin{bmatrix} \tilde{P}_E(t+1) \\ \tilde{P}_I(t+1) \\ \tilde{P}_R(t+1) \end{bmatrix} = \begin{bmatrix} 0_n \\ 0_n \\ P_R^* 1_n \end{bmatrix} + M \begin{bmatrix} \tilde{P}_E(t) \\ \tilde{P}_I(t) \\ \tilde{P}_R(t) - P_R^* 1_n \end{bmatrix},$$

where

$$M = \begin{bmatrix} (1-\epsilon) I_n & (1-\theta) P_S^* \beta A & 0_{n \times n} \\ \epsilon I_n & (1-\delta) I_n & 0_{n \times n} \\ -\theta I_n & (\delta - \theta) I_n & (1-\gamma-\theta) I_n \end{bmatrix}.$$

Chapter 3

IMPROVED BOUNDS ON THE EPIDEMIC THRESHOLD OF THE EXACT MODELS

[1] Navid Azizan et al. "Improved bounds on the epidemic threshold of exact SIS models on complex networks". In: *2016 55th IEEE Conference on Decision and Control (CDC)*. 2016, pp. 3560–3565. DOI: 10.1109/CDC.2016.7798804.

The SIS (susceptible-infected-susceptible) epidemic model on an arbitrary network, without making approximations, is a 2^n-state Markov chain with a unique absorbing state (the all-healthy state). This makes analysis of the SIS model and, in particular, determining the threshold of epidemic spread, quite challenging. We saw in the previous chapter that the exact marginal probabilities of infection can be upper bounded by an n-dimensional linear time-invariant system, a consequence of which is that the Markov chain is "fast-mixing" when the LTI system is stable, i.e., when $\frac{\beta}{\delta} < \frac{1}{\lambda_{\max}(A)}$ (where β is the infection rate per link, δ is the recovery rate, and $\lambda_{\max}(A)$ is the largest eigenvalue of the network's adjacency matrix). This well-known threshold has been recently shown not to be tight in several cases, such as in a star network. In this chapter, we provide tighter upper bounds on the exact marginal probabilities of infection, by also taking pairwise infection probabilities into account. Based on this improved bound, we derive tighter eigenvalue conditions that guarantee fast mixing (i.e., logarithmic mixing time) of the chain. We demonstrate the improvement of the threshold condition by comparing the new bound with the known one on various networks with various epidemic parameters.

3.1 Introduction

The mathematical modeling and analysis of epidemic spread is of great importance in understating dynamical processes over complex networks (e.g., social networks) and has attracted significant interest from different communities in recent years. The study of epidemics plays a key role in many areas beyond epidemiology [29], such as viral marketing [168, 175], network security [8, 2], and information propagation [109, 56]. Although there is a huge body of work on epidemic models, classical ones mostly neglect the underlying network structure and assume a uniformly mixed

population, which is obviously far from reality. However, in recent years, more realistic networked models have been introduced, and many interesting results are now known [155, 165].

In the simplest case (the binary-state or SIS model), each node is in one of two different states: susceptible (S) or infected (I). During any time interval, each susceptible (healthy) node has a chance of being independently infected by any of its infected neighbors (with probability β). Further, during any time interval, each infected node has a chance of recovering (with probability δ) and becoming susceptible again. For a network with n nodes, this yields a Markov chain with 2^n states, which is referred to as the exact or "stochastic" model. Since analyzing this model is quite challenging, most researchers have resorted to n-dimensional linear and nonlinear approximations (the most common being the "mean-field" approximation), which are sometimes called "deterministic" models. This chapter focuses on improving known bounds on the exact model.

The spreading process can be considered either as a discrete-time Markov chain or a continuous-time one. Although the discrete-time model is sometimes argued to be more realistic [79, 4], there is no fundamental difference between the two, and similar results have been shown for both. We focus on the discrete-time Markov chain here.

It is known that these epidemic models exhibit a phase transition behavior at a certain threshold [53, 30] , i.e., once the effective infection rate $\tau = \frac{\beta}{\delta}$ approaches a critical value τ_c [155] the epidemic appears not to die out. We should remark that the Markov chain has a unique absorbing state, which is the all-healthy state, because once the system reaches this state, it remains there forever since there are no infected nodes to propagate infections. This means that if we wait long enough, the epidemic will eventually die out, which may seem to be odd at first. However, what this means is that the question of the epidemic dying out is not interesting; what is interesting is the question of *how long it takes for the epidemic to die out.* In particular, if the mixing time of the Markov chain is exponentially large, one will not see it dying out in any reasonable time. Therefore, the right question to ask is what is the mixing time of the Markov chain (or, equivalently, its mean-time-to-absorption); it turns out that the threshold τ_c corresponds to the phase transition between "slow mixing" (exponential time) and "fast mixing" (logarithmic time) of the MC [64, 204, 205].

The epidemic threshold (critical value) of general networks is still an open problem. However, lower- and upper-bounds have been found using different techniques [64,

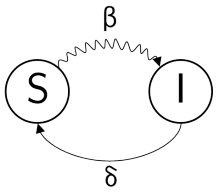


Figure 3.1: State diagram of a single node in the SIS model, and the transition rates. Wavy arrow represents exogenous (neighbor-based) transition. β : probability of infection per infected link, δ : probability of recovery.

205]. The best known *lower-bound* is $1/\lambda_{\max}(A)$, i.e., the inverse of the leading eigenvalue of the adjacency matrix, which is derived by *upper-bounding* the marginal probabilities of infection and using a linear dynamical system. In fact, this method relies on keeping track of n variables which are upper bounds on the marginal probability of infection for any of the nodes. In this chapter, we focus on improving this upper-bound on the infection probabilities and ultimately the lower-bound on the epidemic threshold. The key idea is to maintain the "pairwise" probabilities of nodes' infections, in addition to the marginals. This comes at the cost of increased, yet still perfectly feasible, computation. There is a trade-off between the tightness of the bound and the complexity, and in theory if one takes into account all marginals, pairs, triples, and higher-order terms, we get back to the original 2^n-state Markov chain.

We first briefly review the known bound with marginals, and show a simple alternative approach for deriving it. We then move on to pairs and use the machinery developed in Section 3.2 to derive tighter bounds on the probabilities and connect them with the mixing time of the Markov chain (Sections 3.3 and 3.4). Finally, we demonstrate the improvement of the bounds through extensive simulations (Section 3.5) and conclude with future directions.

3.2 The Markov Chain and Marginal Probabilities of Infection

Let $G = (V, E)$ be an arbitrary connected undirected network with n nodes, and with adjacency matrix A. Each node can be in a state of health, represented by "0," or a state of infection, represented by "1." The state of the entire network can be represented by a binary n-tuple $\xi(t) = (\xi_1(t), \cdots, \xi_n(t)) \in \{0, 1\}^n$, where each of the entries represents the state of a node at time t, i.e., i is infected if $\xi_i(t) = 1$ and it is healthy if $\xi_i(t) = 0$.

Given the current state $\xi(t)$, the infection probability of each node in the next step

is determined independently, and therefore, the transition matrix S of this Markov Chain has elements $S_{X,Y} = \mathbb{P}(\xi(t+1) = Y|\xi(t) = X)$ of the following form:

$$\mathbb{P}(\xi(t+1) = Y|\xi(t) = X) = \prod_{i=1}^{n} \mathbb{P}(\xi_i(t+1) = Y_i|\xi(t) = X), \tag{3.1}$$

for any two state vectors $X, Y \in \{0,1\}^n$.

As mentioned before, a healthy node can receive infection from any of its infected neighbors independently with probability β per infected link, and an infected node can recover from the disease with probability δ. That is

$$\mathbb{P}(\xi_i(t+1) = Y_i|\xi(t) = X) = \begin{cases} (1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (0,0), |N_i \cap \mathbb{S}(X)| = m_i, \\ 1-(1-\beta)^{m_i} & \text{if } (X_i, Y_i) = (0,1), |N_i \cap \mathbb{S}(X)| = m_i, \\ \delta & \text{if } (X_i, Y_i) = (1,0), |N_i \cap \mathbb{S}(X)| = m_i, \\ 1-\delta & \text{if } (X_i, Y_i) = (1,1), |N_i \cap \mathbb{S}(X)| = m_i, \end{cases} \tag{3.2}$$

where $\mathbb{S}(X)$ is the support of $X \in \{0,1\}^n$, i.e., $\mathbb{S}(X) = \{i : X_i = 1\}$, and N_i is the set of neighbors of node i.

Eqs. (3.1, 3.2) completely define the $2^n \times 2^n$ transition matrix of the Markov chain, which determines the evolution of the 2^n states over time. Of course with this, we have the joint probability of all the nodes, and we can compute the probability of any desired combination by marginalizing out the rest. In particular, one can compute the probability of each node i being infected at time $t+1$ (denoted by $p_i(t+1)$), which is a function of all joint probabilities of the states at time t. Since there are only n such variables, the dimension would be significantly reduced if one could "bound" or "approximate" that function by something that includes marginals $p_i(t)$ only. This way, we obtain a recursion which relates the marginals at time $t+1$ to those of time t, and indeed we have a system with only n states rather that 2^n states. Approximations per se are not very interesting because they do not provide any *guarantee* on the behavior of the exact Markov chain model. What is more important is whether one can obtain a bound on these true probabilities, which can guarantee, for example, fast extinction of the disease. The most common upper-bound, which has been shown to be the tightest linear upper-bound with marginals only (using a linear programming technique) [6, 16] is:

$$p_i(t+1) \leq (1-\delta)p_i(t) + \beta \sum_{j \in N_i} p_j(t) \tag{3.3}$$

for all $i = 1, \ldots, n$. Defining $p(t) = (p_1(t), \ldots, p_n(t))^T$, this can be written in a matrix form as

$$p(t+1) \leq Mp(t), \tag{3.4}$$

where

$$M = (1-\delta)I_n + \beta A. \tag{3.5}$$

3.2.1 An Alternative Bounding Technique

The derivation of (3.3) in [6, 16] involves a linear programming technique. In this chapter, we provide an alternative technique to bound the infection probabilities using indicator variables and conditional expectation, which is more intuitive and direct. Importantly, as will be shown later, this technique can be used to obtain tighter bounds on the exact probabilities of infections using pairwise inflectional probabilities. Before that, it is instructive to derive (3.3) using this alternative approach.

Let $i \in V$. We start by conditioning on the state of the same node i at time t, as follows:

$$\begin{aligned} p_i(t+1) =& \mathbb{P}(X_i(t+1) = 1 | X_i(t) = 1)\mathbb{P}(X_i(t) = 1) \\ &+ \mathbb{P}(X_i(t+1) = 1 | X_i(t) = 0)\mathbb{P}(X_i(t) = 0). \end{aligned}$$

The probability that an infected node remains infected is $1-\delta$, and the probability that a susceptible node does not receive infection from an infected neighbor is $1-\beta$. We denote j neighbor of i by $j \sim i$. The expression above can be written as

$$p_i(t+1) = (1-\delta)p_i(t) + \mathbb{E}_{X_{-i}(t)|X_i(t)=0}\left[1 - \prod_{j\sim i}(1-\beta \mathbb{1}_{X_j(t)})\right]\mathbb{P}(X_i(t) = 0). \tag{3.6}$$

The conditional expectation is on the joint probability of all nodes other than i (denoted by X_{-i}), given node i being healthy ($X_i = 0$). Of note, this expression is still exact, and we have not done any approximation yet. It can be easily checked that

$$\prod_{j\sim i}(1-\beta \mathbb{1}_{X_j(t)}) \geq 1 - \beta \sum_{j\sim i} \mathbb{1}_{X_j(t)}.$$

Combining this with (3.6) yields the desired upper bound

$$\begin{aligned} p_i(t+1) &\leq (1-\delta)p_i(t) + \beta \sum_{j\sim i} \mathbb{P}(X_i(t) = 0, X_j(t) = 1) \qquad (3.7) \\ &\leq (1-\delta)p_i(t) + \beta \sum_{j\sim i} p_j(t). \end{aligned}$$

3.2.2 Connection to Mixing Time of the Markov Chain

Up to this point, we just talked about bounding the marginal probabilities of infection, and it is not clear how a bound on the marginal probabilities relates to the mixing time of the Markov chain. To establish this connection, let us start from the definition of mixing time [126]:

$$t_{mix}(\epsilon) = \min\{t : \sup_{\mu} \|\mu S^t - \pi\|_{TV} \leq \epsilon\}, \tag{3.8}$$

where μ is any initial probability distribution defined on the state space, and π is the stationary distribution; $\|\mu - \mu'\|_{TV}$ is the total variation distance of any two probability measures μ and μ', and is defined by

$$\|\mu - \mu'\|_{TV} = \frac{1}{2} \sum_{x} |\mu(x) - \mu'(x)|,$$

where x is any possible state in the probability space. In fact $t_{mix}(\epsilon)$ is the minimum time instant for which the distance between the stationary distribution and the probability distribution at time t from any initial distribution is smaller than or equal to ϵ. Roughly speaking, the mixing time measures how fast the initial distribution converges to the limit distribution, which, in our case, means how quickly the epidemic dies out.

Since, in the stationary distribution, the all-healthy state has probability 1, it can be shown [6] that

$$\sup_{\mu} \|\mu S^t - \pi\|_{TV} = \mathbb{P}\left(\begin{matrix}\text{some nodes are infected at time } t| \\ \text{all nodes were infected at time 0}\end{matrix}\right) \tag{3.9}$$

which highlights the fact that the worst initial distribution (i.e., the μ that maximizes above quantity) is the all-infected state. Now, for any $t < t_{mix}(\epsilon)$ we have

$$\begin{aligned} \epsilon &< \mathbb{P}\left(\begin{matrix}\text{some nodes are infected at time } t| \\ \text{all nodes were infected at time 0}\end{matrix}\right) \\ &\leq \sum_{i=1}^{n} \mathbb{P}\left(\begin{matrix}\text{node } i \text{ is infected at time } t| \\ \text{all nodes were infected at time 0}\end{matrix}\right) \\ &= 1_n^T p(t) \qquad \text{given that } p(0) = 1_n, \end{aligned} \tag{3.10}$$

where we have used the union bound, and 1_n denotes the all-ones vector of size n.

Back to the upper-bound on the marginals (3.4), we get $1_n^T p(t) \leq 1_n^T M p(t-1)$. Furthermore, since M has non-negative entries (we write this as $M \geq 0$), we can "propagate" the bound to find that

$$1_n^T p(t) \leq 1_n^T M p(t-1) \leq 1_n^T M^2 p(t-2) \leq \cdots \leq 1_n^T M^t p(0).$$

As a result, for any $t < t_{mix}(\epsilon)$

$$\epsilon < 1_n^T M^t 1_n \leq n(\rho(M))^t, \tag{3.11}$$

since M is non-negative and symmetric, and $\lambda_{\max}(M) = \rho(M)$, where $\rho(M)$ is the spectral radius of M.

When $\rho(M) < 1$ (or equivalently $1 - \delta + \beta\lambda_{max}(A) < 1$), it follows that $t < \frac{\log \frac{n}{\epsilon}}{-\log \rho(M)}$ for all $t < t_{mix}(\epsilon)$. This implies the well-known result that when $\beta/\delta < 1/\lambda_{\max}(A)$ then $t_{mix}(\epsilon) \leq \frac{\log \frac{n}{\epsilon}}{-\log \rho(M)} = O(\log n)$.

We should note here that if M was not symmetric (as we will encounter such instances in the next section), it can be shown by an appeal to the Lyapunov equation that if $\rho(M) < 1$, then for all $t < t_{mix}(\epsilon)$, there exists $0 < \eta < 1$ such that $\epsilon \leq \eta^t O(\text{poly}(n))$, from which it follows directly that the mixing time is logarithmic in n. To see that, note that $\rho(M) < 1$ implies that there exists a positive definite matrix $P \succ 0$ such that $M^T P M - P \prec 0$. Letting $P^{1/2}$ denote the unique positive square root of P and $N := P^{1/2} M P^{-1/2}$, it follows easily that $N^T N \prec I_d$, or equivalently $\eta := \|N\|_2 < 1$. (Here, $\|N\|_2$ denotes the spectral norm of N.) Defining $y := P^{1/2} 1_n$ and $x := P^{-1/2} 1_n$ we get $1_n^T M^t 1_n = x^T N^t y \leq \|x\|_2 \eta^t \|y\|_2 \leq n\eta^t \|P^{1/2}\|_2 \|P^{-1/2}\|_2$.

3.3 Pairwise Probabilities (p_{ij})

In Section 3.2, we showed how a bound on marginal probabilities of infection can be obtained, and how this bound translates to the threshold condition for fast mixing of the Markov chain. As mentioned before, the bound (3.4) has been proved to be the tightest linear bound one can get with marginals. A natural idea to improve this bound is to go to higher-order terms (i.e., pairs, triples, etc.). In principle, maintaining higher-order terms is advantageous because it means keeping more information from the original chain, but of course at the cost of increased complexity. We define the pairwise probability of infection of two nodes, in addition to the marginals, as follows. For $(i, j) \in E$,

$$p_{ij}(t) := \mathbb{P}(X_i(t) = 1, X_j(t) = 1).$$

Note that out of the $\binom{n}{2}$ possible pairs of nodes, we only consider the ones that correspond to edges in the graph. Based on this definition, $\mathbb{P}(X_i(t) = 0, X_j(t) = 1) = p_j(t) - p_{ij}(t)$, and it follows easily from (3.7) that

$$p_i(t+1) \leq (1-\delta) p_i(t) + \beta \sum_{j \sim i} p_j(t) - \beta \sum_{j \sim i} p_{ij}(t). \tag{3.12}$$

Of course, this bound is at least as tight as the one in (3.3). Now, in order to strictly improve upon the latter, we need a lower bound on the pairwise infection probabilities at time $t+1$ in terms of marginals and pairwise probabilities at time t, which is derived next.

3.3.1 A Lower Bound on the p_{ij}'s

To construct a lower bound on the pairwise marginal probabilities $p_{ij}(t+1)$, we use the same approach as was introduced in Section 3.2.1, but this time applied to pairwise infection probabilities.

Let $(i,j) \in E$ and $t \geq 0$. We first expand $p_{ij}(t+1)$ as follows

$$\sum_{\substack{x\in\{0,1\}\\ y\in\{0,1\}}} \mathbb{P}(X_i(t+1)=1, X_j(t+1)=1, X_i(t)=x, X_j(t)=y).$$

For convenience, denote each one of the summands above by s_{xy}. Also, let c_{xy} represent the corresponding conditional probability $\mathbb{P}(X_i(t+1)=1, X_j(t+1)=1 | X_i(t)=x, X_j(t)=y)$. In what follows, we lower bound each one of s_{xy}'s. We write $\mathbb{E}_{xy}$ for the conditional expectation $\mathbb{E}_{X_{-i,-j}(t)|\{X_i(t)=x, X_j(t)=y\}}$.

• (x=0,y=0): Trivially, $s_{00} \geq 0$.

• (x=0,y=1): As before, the probability of not getting infected from each infected neighbor is $(1-\beta)$, and the probability that an infected node remains infected is $(1-\delta)$. Therefore

$$c_{01} = \mathbb{E}_{01}\big[(1-\prod_{k\sim i}(1-\beta\mathbb{1}_{X_k(t)}))(1-\delta)\big].$$

Since $j \sim i$ and $0 \leq \beta \leq 1$, it follows that $\prod_{k\sim i}(1-\beta\mathbb{1}_{X_k(t)}) \leq (1-\beta\mathbb{1}_{X_j(t)})$. Hence $c_{01} \geq \beta(1-\delta)\mathbb{E}_{01}\mathbb{1}_{X_j(t)}$, which eventually gives

$$\begin{aligned} s_{01} &\geq \beta(1-\delta)\mathbb{P}(X_i(t)=0, X_j(t)=1) \\ &\geq \beta(1-\delta)p_j(t) - \beta(1-\delta)p_{ij}(t). \end{aligned}$$

• (x=1,y=0): By symmetry, the exact same argument as above implies

$$s_{10} \geq \beta(1-\delta)p_i(t) - \beta(1-\delta)p_{ij}(t).$$

• (x=1,y=1): Clearly $c_{11} = (1-\delta)^2$, which gives

$$s_{11} \geq (1-\delta)^2 p_{ij}(t).$$

Summing all the above terms yields the following lower bound for all $(i, j) \in E$ and $t \geq 0$:

$$p_{ij}(t+1) \geq (1-\delta)\beta(p_i(t) + p_j(t)) + (1-\delta)(1-\delta-2\beta)p_{ij}(t). \tag{3.13}$$

3.3.2 Back to the Mixing Time

In order to express Eqs. (3.12) and (3.13) for all i and j's together in a matrix form, recall the definition $p(t) = (p_1(t), \ldots, p_n(t))^T$. Further, let us define $p_E(t) \in \mathbb{R}^{|E|}$ as the vector of pairwise infection probabilities, i.e., $p_E(t) = \text{vec}(p_{ij}(t) : (i, j) \in E)$. Note that $p_{ij}(t) = p_{ji}(t)$, so for each edge we only keep track of one of the two terms. Now we can write Eqs. (3.12), (3.13) as

$$\begin{bmatrix} p(t+1) \\ -p_E(t+1) \end{bmatrix} \leq M' \begin{bmatrix} p(t) \\ -p_E(t) \end{bmatrix}, \tag{3.14}$$

The matrix M', after a little bit of thought, can be expressed in the following way:

$$M' = \begin{bmatrix} (1-\delta)I_n + \beta A & \beta B \\ -(1-\delta)\beta B^T & (1-\delta)(1-\delta-2\beta)I_{|E|} \end{bmatrix}, \tag{3.15}$$

where $B \in \mathbb{R}^{|V| \times |E|}$ happens to be the incidence matrix of G, which is formally defined as

$$B_{i,e} = \begin{cases} 1 & \text{if } i \text{ is an endpoint of } e, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in V$ and $e \in E$.

By accounting for pairwise infection probabilities, the bound derived in (3.14) is tighter when compared to the one in (3.4). In order to connect this to the the mixing time of the underlying Markov chain, observe that

$$1_n^T p(t) = \begin{bmatrix} 1_n^T & 0_{|E|}^T \end{bmatrix} \begin{bmatrix} p(t) \\ -p_E(t) \end{bmatrix}.$$

Applying (3.14) to this gives,

$$1_n^T p(t) \leq \begin{bmatrix} 1_n^T & 0_{|E|}^T \end{bmatrix} M' \begin{bmatrix} p(t-1) \\ -p_E(t-1) \end{bmatrix}. \tag{3.16}$$

This step is possible because the entries of the vector $\begin{bmatrix} 1_n^T & 0_{|E|}^T \end{bmatrix}$ are all non-negative, which guarantees that the signs of all the $n + |E|$ inequalities in (3.14) are preserved.

With this note, it becomes clear that in order to be able to propagate the bounds for the remaining time instances $t-2, t-3, \ldots, 0$, a sufficient condition would be

$$\begin{bmatrix} 1_n^T & 0_{|E|}^T \end{bmatrix} (M')^t \geq 0, \quad \text{for all } t \geq 1. \tag{3.17}$$

Provided that (3.17) holds, we can continue with the sequence of bounds after (3.16), which results in

$$1_n^T p(t) \leq \begin{bmatrix} 1_n^T & 0_{|E|}^T \end{bmatrix} (M')^t \begin{bmatrix} p(0) \\ -p_E(0) \end{bmatrix}. \tag{3.18}$$

Subsequently, the same argument as in Section 3.2.2 concludes the following result.

Theorem 15. *Assume that* (3.17) *holds. If* $\rho(M') < 1$*, then the mixing time of the Markov chain whose transition matrix* S *is described by Eqs.* (3.1) *and* (3.2) *is* $O(\log n)$.

From the standard bound with only marginals, it was known that when $\rho(M) < 1$, the Markov chain is fast-mixing. Now, in addition to that, the above theorem states that when $\rho(M') < 1$, the Markov chain mixes fast again. Of course, this is informative only when there is a case where $\rho(M) > 1$ but $\rho(M') < 1$. As it will be shown in Section 3.5, this is indeed the case.

Note that, in the proof of Theorem 15, we used the assumption that (3.17) holds. As will be shown in the simulations section, in many cases this is a reasonable assumption. However, when the assumption does not hold we cannot appeal to this theorem. For this reason, we propose another bound using an alternative pairwise probability, which does not require such a condition.

3.4 An Alternative Pairwise Probability (q_{ij})

As it was discussed above, when the assumption (3.17) does not hold, we seek an alternative bound. Let us define

$$q_{ij}(t) := \mathbb{P}(X_i(t) = 0, X_j(t) = 1).$$

We can use the same approach as before to obtain bounds for p_i, q_{ij}'s. Intuitively, lower bounding $p_{ij}(t+1)$ is equivalent with upper bounding $q_{ij}(t+1)$, and it turns out that it is what we need. The next lemma summarizes the bounds on these probabilities.

Lemma 16. *For all $i, j \in V$, $(i, j) \in E$ and $t \geq 0$, it holds that*

$$p_i(t+1) \leq (1-\delta)p_i(t) + \beta \sum_{\ell \sim i} q_{i\ell}(t) \tag{3.19}$$

$$q_{ij}(t+1) \leq \delta(1-\delta)p_j(t) + (1-\delta)(1-\delta-\beta)q_{ij}(t) + \beta\delta q_{ji}(t) + \beta(1+\delta) \sum_{\substack{\ell \sim j \\ \ell \neq i}} q_{j\ell}(t) \tag{3.20}$$

Proof. Observe that (3.19) is nothing but (3.12) expressed in terms of q_{ij}'s. Now let $(i, j) \in E$. We first expand $q_{ij}(t+1)$ as follows

$$\sum_{\substack{x \in \{0,1\} \\ y \in \{0,1\}}} \mathbb{P}(X_i(t+1) = 0, X_j(t+1) = 1, X_i(t) = x, X_j(t) = y).$$

For convenience, denote each one of the summands above by s_{xy}. Also, let c_{xy} denote the corresponding conditional probabilities $\mathbb{P}(X_i(t+1) = 0, X_j(t+1) = 1 | X_i(t) = x, X_j(t) = y)$. In what follows, we upper bound each one of the s_{xy}'s; this will immediately yield (3.20). All expectations below are conditional on the event $\{X_i(t) = x, X_j(t) = y\}$, which is omitted for the sake of convenience.

• (x=0,y=0): Using the fact that

$$c_{00} = \mathbb{E}_{00}\Big[\underbrace{\Big(\prod_{k \sim i}(1 - \beta \mathbb{1}_{X_k(t)})\Big)}_{\leq 1} \underbrace{\Big(1 - \prod_{\ell \sim j}(1 - \beta \mathbb{1}_{X_\ell(t)})\Big)}_{\leq \beta \sum_{\ell \sim j} \mathbb{1}_{X_\ell}} \Big],$$

we find that $s_{00} \leq \beta \sum_{\ell \sim j} \mathbb{P}(X_i(t) = 0, X_j(t) = 0, X_\ell(t) = 1) \leq \beta \sum_{\substack{\ell \sim j \\ \ell \neq i}} q_{j\ell}(t)$.

• (x=0,y=1): From

$$c_{01} = \mathbb{E}_{01}\Big[\Big(\prod_{k \sim i}(1 - \beta \mathbb{1}_{X_k(t)})\Big)(1-\delta)\Big] \leq (1-\beta)(1-\delta),$$

it follows that $s_{01} \leq (1-\beta)(1-\delta)q_{ij}(t)$.

• (x=1,y=0): Using the fact that

$$c_{10} = \mathbb{E}_{10}\Big[\delta\Big(1 - \prod_{\ell \sim j}(1 - \beta \mathbb{1}_{X_\ell(t)})\Big)\Big] \leq \mathbb{E}_{10}\Big[\beta\delta \sum_{\ell \sim j} \mathbb{1}_{X_\ell}\Big],$$

we find that $s_{10} \leq \beta\delta \sum_{\ell \sim j} \mathbb{P}(X_i(t) = 1, X_j(t) = 0, X_\ell(t) = 1) \leq \beta\delta \sum_{\ell \sim j} q_{j\ell}(t)$.

• (x=1,y=1): Using the fact that $c_{11} = \delta(1-\delta)$, we find that $s_{11} = \delta(1-\delta)\mathbb{P}(X_i(t) = 1, X_j(t) = 1) = \delta(1-\delta)(p_j - q_{ij})$.

□

Similar as before, by defining a vector $q_E(t) \in \mathbb{R}^{2|E|}$ as $q_E(t) = \text{vec}(q_{ij}(t) : (i,j) \in E)$, we can express (3.19) and (3.20) as

$$\begin{bmatrix} p(t+1) \\ q_E(t+1) \end{bmatrix} \leq M'' \begin{bmatrix} p(t) \\ q_E(t) \end{bmatrix}, \tag{3.21}$$

for some appropriately defined square matrix M'' of size $n + 2|E|$. It is easy to see that if $1 - \delta - \beta \geq 0$, then $M'' \geq 0$, i.e., all entries of M are nonnegative. In particular, this implies that M'' satisfies (3.17), and it only takes repeating the same argument as in (3.18) to conclude with the following theorem.

Theorem 17. *If $\rho(M'') < 1$ and $1 - \delta - \beta \geq 0$, then the mixing time of the Markov chain whose transition matrix S is described by Eqs.* (3.1) *and* (3.2) *is $O(\log n)$.*

3.5 Experimental Results

In this section, we demonstrate the performance of the proposed bounds by evaluating them on a variety of networks such as clique, Erdős-Rényi, Watts-Strogatz, star graph, line graph, cycle, and star-line graph, with various parameters β and δ. As mentioned before, in order for any of the two threshold conditions proposed in Sections 3.3 and 3.4 to be an improvement, we need to check if there are cases where the spectral radius of M' or M'' is *less than* 1, while the spectral radius of M is *greater than* 1 (or equivalently $\frac{\beta\lambda_{\max}(A)}{\delta} > 1$). Indeed, extensive simulations on our first bound (M') suggest that not only are there such cases, but interestingly always $\rho(M') \leq \rho(M)$.

In order to compare M' with M, we set $\rho(M) = 1 + \epsilon > 1$ for some small value of ϵ, and observe the value of $\rho(M')$. Table 3.1 lists the values of spectral radii for the three matrices. The positive sign next to $\rho(M')$ indicates that the non-negativity condition (3.17) holds. For the cases that the condition holds (+), we can conclude that M' has clearly improved. For the cases where the condition does not hold (-) we evaluate the second proposed bound M'', which again shows clear improvement over the $\rho(M) < 1$ condition.

In order to demonstrate how tight the new condition is, Fig. 3.2 plots the evolution of the epidemic over a star graph, for which the $\rho(M) < 1$ condition is known not to be tight. The parameters in the two cases are $\delta = 0.3, \beta = 0.0130$, and $\delta = 0.3, \beta = 0.0157$. It can be seen that while the value of $\frac{\beta\lambda_{\max}(A)}{\delta}$ is not informative (it is $1.93 > 1$ for the first case, and $2.33 > 1$ for the second one), the $\rho(M')$ condition is quite tight.

Table 3.1: Performance of the proposed bounds M' and M'' in comparison with the previous bound M. Boldface values show an improvement over M. The signs next to $\rho(M')$ indicate whether the non-negativity condition (required for the proof) holds.

		$\rho(M) = 1+\epsilon$	$\rho(M')$		$\rho(M'')$
Star	δ=0.750, β=0.078	1.030	**0.903**	+	
	δ=0.500, β=0.053	1.030	**0.828**	+	
	δ=0.250, β=0.028	1.030	0.840	-	**0.968**
Cycle	δ=0.750, β=0.390	1.030	**0.817**	+	
	δ=0.500, β=0.265	1.030	0.720	-	**0.945**
	δ=0.250, β=0.140	1.030	0.882	-	**0.942**
Star-line	δ=0.750, β=0.174	1.030	**0.872**	+	
	δ=0.500, β=0.118	1.030	0.693	-	**0.958**
	δ=0.250, β=0.063	1.030	0.856	-	**0.955**
Clique	δ=0.750, β=0.008	1.003	**0.999**	+	
	δ=0.500, β=0.005	1.003	**0.998**	+	
	δ=0.250, β=0.003	1.003	**0.999**	+	
Erdős-	δ=0.750, β=0.070	1.030	**0.993**	+	
Rényi	δ=0.500, β=0.048	1.030	**0.977**	+	
	δ=0.250, β=0.026	1.030	**0.984**	+	
Watts-	δ=0.750, β=0.077	1.030	**0.991**	+	
Strogatz	δ=0.500, β=0.053	1.030	**0.974**	+	
	δ=0.250, β=0.028	1.030	**0.982**	+	

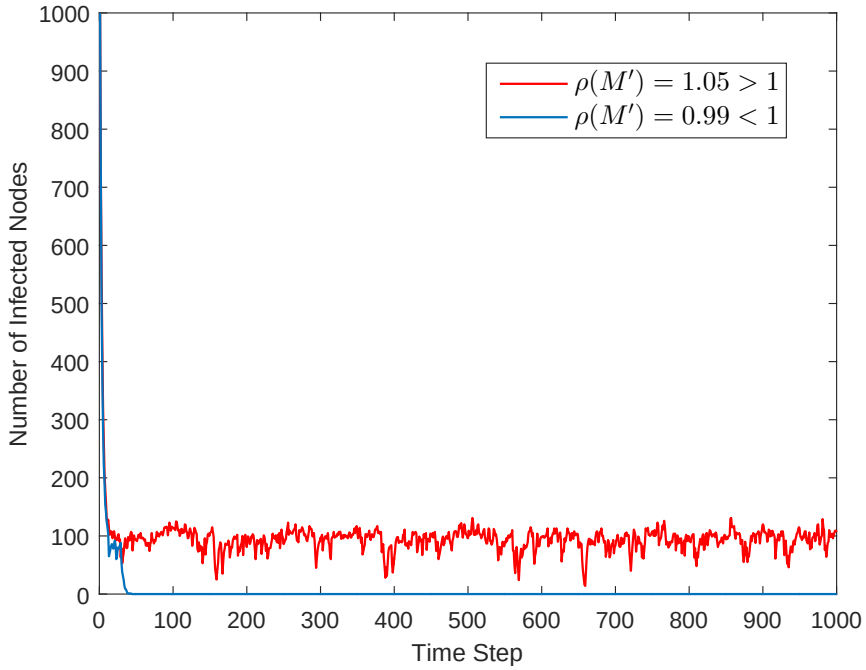


Figure 3.2: Evolution of the SIS epidemic over a star graph with $n = 2000$ nodes, with two values of $\rho(M')$ below and above 1. When $\rho(M') = 0.99 < 1$, we observe fast extinction of the epidemic (blue curve). The condition also seems very tight, as for $\rho(M') = 1.05 > 1$, the epidemic does not die out (red curve). This is while the previously known bound is not informative at all ($\frac{\beta\lambda_{\max}(A)}{\delta} = 1.93 > 1$ for the first case, and $2.33 > 1$ for the second one).

3.6 Conclusion and Future Work

In this chapter, we first proposed a simple technique using conditional expectations to systematically construct bounds on the exact probabilities of infection, up to any desired order. Using this approach, we showed that keeping higher-order terms (such as pairs) indeed helps in obtaining tighter bounds; specifically, we derived a bound composed of both marginals and pairwise probabilities, which has improved over the well-known bounds. Based on this new bound, we provided a new condition for fast mixing of the Markov chain to the all-healthy state, which, through extensive simulations, was shown to be tighter than the so-called $\frac{\beta\lambda_{\max}(A)}{\delta} < 1$ condition.

Clearly, one possible extension of this work would be to construct a bound consisting of marginals, pairs, and triples, which in theory should result in an improvement. In fact, keeping all higher-order terms eventually takes us back to the 2^n-state Markov chain. Therefore, there is a trade-off between the complexity and the accuracy of the model. We should however note that going to triples may still be tractable, and one advantage of that would be not only to gain by improving the bound on the probability

(same as here) but also to get an improvement in the fast-mixing condition of the chain in the sense that the bound (3.10) can be replaced by $\epsilon < \sum_i p_i - \sum_{i,j} p_{ij} + \sum_{i,j,k} p_{ijk}$, which naturally includes all terms rather than just the marginals. As a last comment, based on the simulations, we conjecture that condition (3.17) may be relaxed, and other future work may concern its proof.

Part II

Incentives and Markets

Chapter 4

OPTIMAL PRICING IN MARKETS WITH NON-CONVEX COSTS

[1] Navid Azizan et al. "Optimal Pricing in Markets with Non-Convex Costs". In: *Proceedings of the 2019 ACM Conference on Economics and Computation (EC)*. Phoenix, AZ, USA, 2019, p. 595. ISBN: 978-1-4503-6792-9. DOI: 10.1145/3328526.3329575.

[2] Navid Azizan et al. "Optimal Pricing in Markets with Nonconvex Costs". In: *Operations Research* 68.2 (2020), pp. 480–496. DOI: 10.1287/opre.2019.1900.

Designing markets and incentives in large-scale systems is crucial for efficient operation of the system. In this chapter, we consider a market run by an operator, who seeks to satisfy a given consumer demand for a commodity by purchasing the needed amount from a group of competing suppliers with non-convex cost functions. The operator knows the suppliers' cost functions and announces a price/payment function for each supplier, which determines the payment to that supplier for producing different quantities. Each supplier then makes an individual decision about how much to produce, in order to maximize its own profit. The key question is how to design the price functions. To that end, we propose a new pricing scheme, which is applicable to general non-convex costs, and allows using general parametric pricing functions. Optimizing for the quantities and the price parameters simultaneously, and the ability to use general parametric pricing functions, allows our scheme to find prices that are typically economically more efficient and less discriminatory than those of the existing schemes. In addition, we supplement the proposed method with a polynomial-time approximation algorithm, which can be used to approximate the optimal quantities and prices. Our framework extends to the case of networked markets, which, to the best of our knowledge, has not been considered in previous work.

4.1 Introduction

While there has been a long history of studying markets under convexity assumptions (such as convexity of cost functions, preferences, etc.) in economic theory, *non-convexities* are ubiquitous in most real-world markets. Non-convexities in cost

functions arise due to start-up or shut-down costs, indivisibilities, avoidable costs, or simply economies of scale.

It has been widely noted in the literature that in the presence of non-convexities, there may be no linear prices (constant per quantity) that support a competitive market equilibrium [e.g., 46, 83], and it was suggested as early as the 1980s that in these markets, one needs to consider using *price functions*, as opposed to the conventional prices [212]. Following the work of [178, 179], there have been many pricing schemes proposed in the literature. In particular, during the past decade, motivated by the deregulation of the electricity markets in the US and around the world, the problem of pricing in non-convex markets has attracted renewed interest from researchers, and there has been considerable work on this problem [131]. These pricing schemes are deployed in practice, and the operation and efficiency of our electricity markets relies on them.

Formally, the non-convex pricing problem is that, given an inelastic demand for a commodity from a number of consumers, a market operator seeks to satisfy the demand by purchasing the required amount from a group of competing suppliers with non-convex cost functions. The operator knows the suppliers' cost functions, and it announces a price/payment function for each supplier, which determines the payment to that supplier for producing different quantities. Each supplier then makes an individual decision about how much to produce in order to maximize its own profit. The key design question is how to devise the price functions in order to ensure certain economic properties for the market. We should remark that this problem is quite different from mechanism design, since the cost functions of the suppliers are known to the market operator, and the players can influence the market only by choosing their production level. However, as we shall see, the design of the price functions in these markets is challenging.

An important early approach to the pricing problem was the work of [159], sometimes referred to as integer programming (IP) pricing, which considered the class of non-convexities that arise from the start-up costs of the suppliers (with linear marginal costs). The paper proposes a clever pricing rule, based on solving a mixed-integer linear program and forcing the integral variables to their optimal values as a constraint. The scheme is economically efficient and has a nice dual interpretation. Modified versions of IP pricing have been proposed by [39, 38] and others. Another approach, proposed for the more general class of non-convex cost functions that are in the form of a start-up plus a convex (rather than linear) cost, is the minimum-uplift (MU)

pricing of [103], and its closely related refinement of [81], known as convex hull (CH) pricing. These schemes provide discriminatory uplifts to different suppliers to incentivize production, and the uplifts are minimal in a specific sense [181]. The possibility of having both positive and negative uplifts was also considered by [145, 75]. Other pricing schemes include the semi-Lagrangian relaxation (SLR) approach of [10] and the primal-dual (PD) approach of [177]. These schemes seek to find uniform linear prices that are revenue-adequate (but not supporting of the equilibrium). A survey on all the above pricing schemes was recently written by [131]. However, the overall desired properties, as well as the properties that each of the schemes satisfy, were not examined there. We formalize the desired properties considered in the literature in Section 4.2 and discuss the properties of the existing schemes in Section 4.5. Table 4.1 summarizes the common schemes and their properties.

Despite the large body of work on the pricing problem, the existing schemes have several shortcomings. For example, most of the existing schemes mentioned above are proposed for specific classes of non-convex cost functions and cannot handle more general non-convexities. Furthermore, even the ones that are applicable for general cost functions fail to satisfy some of the key desired properties of the market, such as economic efficiency or supporting a competitive equilibrium. In addition, none of the existing schemes is accompanied by a computationally tractable algorithm for general non-convexities, and they typically rely on off-the-shelf heuristic solvers for mixed-integer programs that are known to be NP-hard.

In this chapter, we propose a pricing scheme for markets with general non-convex costs that designs arbitrary parametric price functions and addresses all the issues mentioned above. Our approach seeks to find the optimal schedule (allocation) and the optimal pricing rule simultaneously, which generally allows for finding *economically more efficient* solutions. The ability to use arbitrarily specified parametric price functions (e.g., piece-wise linear, quadratic, etc.) enables our approach to design price functions that are *less discriminatory*, while still *supporting a competitive equilibrium*. Furthermore, our pricing scheme is accompanied by a *computationally efficient* (polynomial-time) approximation algorithm which allows one to find the approximately-optimal schedule and prices for general non-convex cost functions. Lastly, we extend the proposed pricing rule to *networked* markets, which, to the best of our knowledge, are not considered in any of the existing work.

Specifically, this chapter makes the following contributions.

1. We propose a framework for pricing in markets with *general* non-convex costs, using *general* price functions (Section 4.3.1). Our scheme seeks to find the optimal price functions and allocations simultaneously, while imposing the equilibrium conditions as constraints. For this reason, our approach is generally economically more efficient than the existing methods, while satisfying the equilibrium conditions. Moreover, the ability to use general price forms allows one to obtain more uniform prices (smaller "uplifts").

2. We supplement our pricing scheme with a computationally efficient (polynomial-time) approximation algorithm for finding the allocations and prices (Section 4.3.2).

3. We extend our framework to *networked* markets, and also propose an approximation algorithm that can compute the solution efficiently for acyclic networks, a common scenario in electric distribution networks (Section 4.4).

4. We survey the common pricing schemes proposed in the literature for markets with non-convex costs and provide a compact summary of their properties (Section 4.5).

5. We evaluate the proposed method through extensive numerical examples and show how it compares with the existing methods (Section 4.6).

4.2 Market Description and Pricing Objectives

While our goal in this chapter is to design an economically and computationally efficient pricing scheme for non-convex *networked* markets, we begin with the problem of designing one for a *single* market, which is difficult in its own right. We return to the case of networked markets in Section 4.4. When the cost functions are non-convex, even this seemingly simple problem has proven to be challenging, and a wide variety of pricing schemes have been proposed for it in the literature. In the following, we describe the market model and survey the desired market properties.

4.2.1 Market Model

We consider a single market consisting of n competing suppliers (often referred to as firms or generators). The market is run by a market operator that seeks to satisfy a deterministic and inelastic demand d for a commodity in a single period. Each supplier i has a cost function $c_i(q_i)$ for producing quantity q_i, which may be non-convex.

The suppliers' cost functions are known by the operator, and the operator uses them to determine the prices. In particular, the operator announces price/payment functions $p_i(q_i)$, which determine the payment to supplier i when producing q_i. Note that, in general, the price functions can be different for different suppliers, but it is often desired to have close-to-uniform prices.

Upon the announcement of the price functions, each supplier i makes an individual decision, based on the price function $p_i(\cdot)$ and the cost function $c_i(\cdot)$, about how much to produce (and whether to participate in the market), in order to maximize its own profit, i.e., $p_i(q_i) - c_i(q_i)$. The suppliers are then paid for their production according to the payment function, and the demand (consumers) is charged for the total payment.

This model is classical, and has been studied in a wide variety of contexts, initially under the assumption of convex cost functions for production and linear pricing functions, but more recently under non-convex cost functions. Non-convex cost functions are particularly important in the context of electricity markets. As a result, there is a large literature focusing on non-convex pricing in electricity markets (see [131] for a recent survey). Often this literature assumes specific forms of non-convexities (e.g., startup/fixed costs) and specific forms of payment functions (e.g., linear plus uplift). The results from this literature have guided the design and operation of electricity markets across the world today.

4.2.2 Pricing Objectives

The key design question in the market described above is how to devise the payment functions. The goal of the operator is to (1) find the optimal quantities q_i^*, and (2) design the payment functions $p_i(\cdot)$ that ensure that the suppliers produce the optimal quantities q_i^*.

There is a huge design space for such payment functions, and there is a large literature evaluating proposals in the context of non-convex cost functions, e.g., [159, 103, 39, 81, 10, 177, 131, 181, 107].

From this literature has emerged a variety of desirable properties which pricing rules attempt to satisfy. The following is a summary of the most sought-after properties in this literature. Note that no existing rules satisfy all of these properties for general non-convex markets.

1. ***Market Clearing*** (a.k.a. ***Load Balancing***): The total supply is equal to the demand, i.e., $\sum_{i=1}^{n} q_i^* = d$.

2. ***Economic Efficiency***

    a) ***Minimal Production Cost*** *(Suppliers' Total Cost)*: The total production cost of the suppliers, i.e., $\sum_{i=1}^{n} c_i(q_i^*)$, is minimal.

    b) ***Minimal Payment*** *(Total Paid Cost)*: The total cost that is paid to the suppliers for the commodity, i.e., $\sum_{i=1}^{n} p_i(q_i^*)$, is minimal.

3. ***Incentivizing***

    a) ***Revenue Adequacy***: For every supplier, the net profit at the optimum is nonnegative, i.e., $p_i(q_i^*) - c_i(q_i^*) \geq 0$, for $i = 1, \ldots, n$.

    b) ***Support a Competitive Equilibrium***: The optimum production quantity for each supplier is a maximizer of its individual profit, i.e., $q_i^* \in \arg\max_{q_i} \; p_i(q_i) - c_i(q_i)$, or equivalently $p_i(q_i^*) - c_i(q_i^*) \geq \max_{q_i \neq q_i^*} \; p_i(q_i) - c_i(q_i)$, for $i = 1, \ldots, n$.

4. ***Simplicity and Uniformity***: The price functions are simple and interpretable (ideally linear: $p_i(q_i) = \lambda_i q_i$) and non-discriminatory (ideally uniform across suppliers: $p_i(q_i) = p(q_i)$).

5. ***Computational Tractability***: The optimal quantities and price functions can be computed/approximated in time that is polynomial in n.

A few remarks about these properties are warranted. Property 1 ensures that the demand is met. Property 2 is somewhat more elaborate and concerns the economic efficiency of the scheme, in terms of total expenditure. Even though in certain cases (e.g., in IP pricing of [159] for startup-plus-linear costs), the suppliers' total cost $\sum_{i=1}^{n} c_i(q_i)$ and the total paid cost $\sum_{i=1}^{n} p_i(q_i)$ match and are both minimal, there is an inevitable gap between the two in general. Ultimately, the quantity which determines the cost of satisfying the demand is the total payment to the suppliers $\sum_{i=1}^{n} p_i(q_i)$, and therefore Property 2b is arguably more crucial than Property 2a. However, ostensibly, because the price functions are not directly available while computing the optimal quantities, many pricing schemes have resorted to minimizing the total suppliers' cost $\sum_{i=1}^{n} c_i(q_i)$ as a surrogate for the paid cost. In this work, we advocate a direct approach for minimizing the total payment.

Property 3 incentivizes the suppliers to follow the dispatch and produce the socially-optimal quantities. More specifically, Property 3a ensures that the suppliers do not lose by producing q_i^*, and further, Property 3b assures that it is in each supplier's best interest to follow the dispatch. Property 3b is generally a stronger condition than Property 3a, and if $p_i(0) = c_i(0) = 0\ \forall i$, then (3b) implies (3a).

Property 4 concerns having price forms that are "close to linear" (simple) and "close to uniform" (non-discriminatory), in some sense. One common approach to this is to use uniform linear prices plus a generator-dependent "uplift," i.e., $p_i(q_i) = \lambda q_i + u_i \mathbb{1}_{q_i = q_i^*}$, and try to minimize the uplifts u_i. As Property 4 is subjective by its nature, we allow arbitrary parametrized price functions in our scheme. However, we also examine our scheme when applied to the popular minimal-uplift approach. Note that Property 4 also rules out the use of "dictatorial" prices, in which the operator pays the cost (plus an ϵ) only at the desired amount, and pays nothing for any other amount produced.

The final property, Computational Tractability, is particularly challenging to address in the context of non-convex markets. Nearly all standard approaches work by computing the optimal production quantities and then deriving the prices from these quantities in some way. Under general non-convex cost functions, this first step is already computationally intractable. Thus, it is important to consider relaxations (approximations) of other properties if the goal is to enforce computational tractability. To that end, we consider approximate versions of the Incentivizing and Economic Efficiency conditions, which we discuss in Section 4.3.2. We propose an algorithm that satisfies these approximate versions, while being computationally tractable.

4.3 Proposed Scheme: Equilibrium-Constrained Pricing

Most existing schemes in the literature (see Section 4.5 for a detailed summary) are proposed for specific classes of non-convexities, and are not applicable for more general non-convex costs. Furthermore, even the ones that are applicable for more general cost functions either already lack some of the key properties (such as economic efficiency) or they lose those properties for more general costs. Additionally, the existing schemes are not accompanied by a computationally tractable algorithm for general non-convexities, and they typically rely on off-the-shelf heuristic solvers for mixed-integer programs that are NP-hard. This serves to emphasize that no existing pricing scheme satisfies the desired properties described in Section 4.2.2.

The main contribution of this chapter is the introduction of a new pricing scheme, *Equilibrium-Constrained (EC) pricing*, which is applicable to general non-convex

costs, allows using general parametric price functions, and satisfies all the desired properties outlined before, as long as the price class is general enough. The name of this scheme stems from the fact that we directly impose all the equilibrium conditions as constraints in the optimization problem for finding the best allocations, as opposed to adjusting the prices later to make the allocations an equilibrium. The optimization problem is, of course, non-convex, and non-convex problems are intractable in general. However, we also present a tractable approximation algorithm for approximately solving the proposed optimization.

We present the formulation of the optimization at the core of Equilibrium-Constrained pricing in Section 4.3.1, and then develop an efficient algorithm for solving the optimization problem approximately in Section 4.3.2.

4.3.1 Pricing Formulation

In this section, we propose a systematic approach for determining a pricing rule under generic non-convex costs that minimizes payments and satisfies the properties outlined in Section 4.2.2, while allowing flexibility in the choice of the form of price functions.

Specifically, consider a class of desired price functions, denoted by $\mathcal{P}$, which can be an arbitrary class such as linear, linear plus uplift, piece-wise linear, etc. This choice can be due to interpretability/uniformity reasons or other practical considerations. The core of Equilibrium-Constrained pricing is an optimization problem for finding the best price functions in $\mathcal{P}$ and the best allocations at the same time. The operator is buying the commodity from the suppliers, on behalf of the consumers, and therefore its objective is to minimize the total cost incurred (total payment), subject to the equilibrium constraints. The optimization problem can be expressed as follows.

Equilibrium-Constrained (EC) Pricing:

$$p^* = \min_{\substack{q_1,\dots,q_n \\ p_1,\dots,p_n \in \mathcal{P}}} \quad \sum_{i=1}^{n} p_i(q_i) \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d \tag{4.1b}$$

$$p_i(q_i) - c_i(q_i) \geq 0, \quad i = 1, \dots, n \tag{4.1c}$$

$$p_i(q_i) - c_i(q_i) \geq \max_{q_i' \neq q_i} \; p_i(q_i') - c_i(q_i'), \quad i = 1, \dots, n \tag{4.1d}$$

Constraints (4.1b), (4.1c), and (4.1d) are the Market Clearing, Revenue Adequacy,

and Competitive Equilibrium conditions, respectively. Constraint (4.1d) can also be equivalently expressed as

$$p_i(q_i) - c_i(q_i) \geq p_i(q_i') - c_i(q_i') \quad \forall q_i' \neq q_i, \quad i = 1, \ldots, n. \tag{4.2}$$

The key difference between EC pricing and the existing methods for pricing in non-convex markets is that it directly minimizes the total paid cost and seeks to find both the optimal allocations q_i^* and the optimal price functions $p_i^*(.)$ simultaneously. The scheme enforces the desired economic properties as constraints, while allowing the use of any class of price functions, rather than imposing a fixed form for the price.

Since this scheme minimizes the total payments, and does not impose any explicit constraint on the total production cost, it would be natural to ask what happens to latter quantity as we minimize the former. The minimum total production cost is defined as $c^* = \sum_{i=1}^n c_i(q_i^0)$, where

$$(q_1^0, \ldots, q_n^0) = \underset{q_1,\ldots,q_n}{\arg\min} \quad \sum_{i=1}^n c_i(q_i) \tag{4.3a}$$

$$\text{s.t.} \quad \sum_{i=1}^n q_i = d \tag{4.3b}$$

is the "minimal production cost" solution.

Remark. *It is easy to see, by relaxing the last constraint* (4.1d)*, and using constraint* (4.1c)*, that the optimal value of the optimization problem* (4.1) *is bounded below by the minimum total production cost. Mathematically, we have*

$$\begin{aligned} p^* \geq \quad & \min_{\substack{q_1,\ldots,q_n \\ p_1,\ldots,p_n}} \sum_{i=1}^n p_i(q_i) && \geq \min_{q_1,\ldots,q_n} \sum_{i=1}^n c_i(q_i) \qquad = c^* \\ & \textit{s.t.} \quad \sum_{i=1}^n q_i = d && \textit{s.t.} \quad \sum_{i=1}^n q_i = d \\ & \quad p_i(q_i) \geq c_i(q_i), \quad i = 1, \ldots, n \end{aligned}$$

In other words, the total production cost is always upper-bounded by the total payment. Therefore, minimizing the total payment puts a cap on the total production cost as well, while the opposite is not true in general (minimizing the total production cost can result in very high payments, which can be seen in, e.g., the case studies in Figs. 4.4a and 4.5a).

Remark. *We have imposed nearly all the desired properties as constraints in the optimization problem* (4.1)*, and it might not be clear whether this optimization problem has a solution at all. Indeed, there always exists a class of price functions for which problem* (4.1) *has a solution, and further, the bound mentioned in Remark 4.3.1 is achieved.*

A naive choice of price function, often referred to as dictatorial pricing, is enough to prove this claim. In fact, one can check that for any price function of the form

$$p_i(q_i) \begin{cases} = c_i(q_i) & \text{for } q_i = q_i^0 \\ \leq c_i(q_i) & \text{for } q_i \neq q_i^0 \end{cases},$$

problem (4.1) *has an optimal solution $q^* = q^0$, and it achieves the bound $p^* = c^*$.*

While Remark 4.3.1 asserts the existence of an optimal price function in general, the problem may not have a solution for certain specific classes of price functions. The key point is that problem (4.1) always allows using more sophisticated price forms (e.g., piece-wise linear) for which it will have a solution; and for any given choice of price form, it finds the best one, along with the optimal quantities.

Remark. *While in most scenarios, the operator is buying the commodity from the suppliers on behalf of the consumers, and it makes sense to minimize the total payments $\sum_{i=1}^n p_i(q_i)$, in general one may seek to balance between the consumers' and the suppliers' costs. In other words, one can take the objective to be a linear combination of the consumers' cost $\sum_{i=1}^n p_i(q_i)$ and the suppliers' net cost (negative profit) $\sum_{i=1}^n (c_i(q_i) - p_i(q_i))$. Without loss of generality, the weighted sum can be normalized to an affine (i.e., convex) combination $(1-\theta)\sum_{i=1}^n p_i(q_i) + \theta \sum_{i=1}^n (c_i(q_i) - p_i(q_i))$ with parameter θ. The optimization can be expressed as follows.*

$$p_\theta^* = \min_{\substack{q_1,\ldots,q_n \\ p_1,\ldots,p_n \in \mathcal{P}}} \quad (1-2\theta)\sum_{i=1}^n p_i(q_i) + \theta \sum_{i=1}^n c_i(q_i) \tag{4.4a}$$

$$\text{s.t.} \quad \sum_{i=1}^n q_i = d \tag{4.4b}$$

$$p_i(q_i) - c_i(q_i) \geq 0, \quad i = 1, \ldots, n \tag{4.4c}$$

$$p_i(q_i) - c_i(q_i) \geq \max_{q_i' \neq q_i} p_i(q_i') - c_i(q_i'), \quad i = 1, \ldots, n \tag{4.4d}$$

For the cases when the total payment $p^* = \sum_{i=1}^n p_i(q_i^*)$ *from the optimization problem* (4.1) *matches the lower bound* $c^* = \sum_{i=1}^n c_i(q_i^*)$ *(such as in the linear+uplift example of Section 4.3.1.1), the solution from* (4.4) *is the same as that of* (4.1)*, and the prices will be insensitive to parameter* θ*.*

It is worth mentioning that our algorithm proposed in Section 4.3.2 for solving (4.1) *is also capable of handling the weighted problem* (4.4)*. However, for the sake of simplicity, we focus on the case of* $\theta = 0$*.*

To be more explicit about the class of price functions, we consider a general parametric form for $\mathcal{P}$, specified by $p_i(q_i) := p(q_i; \alpha, \beta_i)$ with two types of parameters $\alpha \in \mathbb{R}^{l_1}$, and $\beta_i \in \mathbb{R}^{l_2}$ for $i = 1, \ldots, n$, where parameter α is shared among all the suppliers, and it constitutes the uniform component of the price, while parameter β_i is specific to supplier i. The parameters are in general constrained to be in some bounded sets $\mathcal{A} \subseteq \mathbb{R}^{l_1}$ and $\mathcal{B} \subseteq \mathbb{R}^{l_2}$, i.e., $\alpha \in \mathcal{A}$, and $\beta_i \in \mathcal{B}$ for all $i = 1, \ldots, n$. This parametric form is general enough that it encompasses all the assumed price forms in the literature. In particular, the linear-plus-uplift form ($p_i(q_i) = \lambda q_i + u_i \mathbb{1}_{q_i = \hat{q}_i}$) is a special case of this form, where the shared parameter is the uniform price λ, and the individual parameters are the amount and location of the uplifts $u_i, \hat{q}_i$. Using the general parametric form, the optimization problem (4.1) can be re-expressed as follows.

Parameterized Equilibrium-Constrained (EC) Pricing:

$$p^* = \min_{\substack{q_1,\ldots,q_n \\ \alpha \in \mathcal{A} \\ \beta_1,\ldots,\beta_n \in \mathcal{B}}} \quad \sum_{i=1}^n p(q_i; \alpha, \beta_i) \tag{4.5a}$$

$$\text{s.t.} \quad \sum_{i=1}^n q_i = d \tag{4.5b}$$

$$p(q_i; \alpha, \beta_i) - c_i(q_i) \geq 0, \quad i = 1, \ldots, n \tag{4.5c}$$

$$p(q_i; \alpha, \beta_i) - c_i(q_i) \geq \max_{q_i' \neq q_i} \; p(q_i'; \alpha, \beta_i) - c_i(q_i'), \quad i = 1, \ldots, n \tag{4.5d}$$

To show a concrete application of this general pricing scheme, we apply our framework to the popular class of linear-plus-uplift price functions, which has been a standard form considered in the electricity markets literature [e.g., in 103, 81], and minimize the uplifts. We derive closed-form solutions for the optimal quantities and prices

(for general cost functions). In this case, the total payment matches the total cost, which is the lowest theoretically possible. In contrast, the convex hull (CH) and minimum-uplift (MU) pricing schemes, which are the most closely related schemes and use the same type of price functions fail to achieve this bound and typically exhibit a large gap. The integer programming (IP) pricing, on the other hand, is capable of achieving the bound, but only for startup+linear cost functions, and not for more general cost functions such as startup+convex. (See Section 4.5 for more details on the existing schemes and their comparison with EC.)

4.3.1.1 Linear+Uplift Pricing

As mentioned earlier, using a linear uniform price plus an uplift term is a common choice of class of price functions, in practice. For this class, we have $p(q_i; \lambda, u_i, \hat{q}_i) = \lambda q_i + u_i \mathbb{1}_{q_i=\hat{q}_i}$, where $\lambda, u_1, \ldots, u_n \geq 0$. Without loss of generality, we can assume $\hat{q}_i^* = q_i^*$, i.e., the optimal location of uplift coincides with the desired production level, which is intuitive (see the appendix for proof). The optimization problem (4.5) can then be reduced to

$$p^*_{\text{uplift}} = \min_{\substack{q_1,\ldots,q_n \\ \lambda \geq 0 \\ u_1,\ldots,u_n \geq 0}} \quad \sum_{i=1}^{n} (\lambda q_i + u_i) \tag{4.6a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d \tag{4.6b}$$

$$\lambda q_i + u_i - c_i(q_i) \geq 0, \quad i = 1, \ldots, n \tag{4.6c}$$

$$\lambda q_i + u_i - c_i(q_i) \geq \max_{q_i' \neq q_i} \lambda q_i' - c_i(q_i'), \quad i = 1, \ldots, n \tag{4.6d}$$

Remark. *From Remark 4.3.1, we know that $p^*_{uplift} \geq c^*$. On the other hand, plugging the feasible point $\left(q_i = q_i^0 \ \forall i,\ \lambda = 0,\ u_i = c_i(q_i^0) \ \forall i\right)$ into* (4.6) *results in $p^*_{uplift} \leq c^*$. Therefore $p^*_{uplift} = c^*$.*

Problem (4.6) has potentially many solutions, and the solution $q_i = q_i^0 \ \forall i,\ \lambda = 0,\ u_i = c_i(q_i^0) \ \forall i$ corresponds to the naive pay-as-bid scheme, which is equivalent to having no uniform price and paying each supplier for its own cost. To obtain price functions that are close to uniform, it is desirable to pick a solution for which the uplifts are minimum (in ℓ_1 sense, for example). That is equivalent to adding a layer on top of the optimization problem (4.6) to pick the minimal-uplift solution among

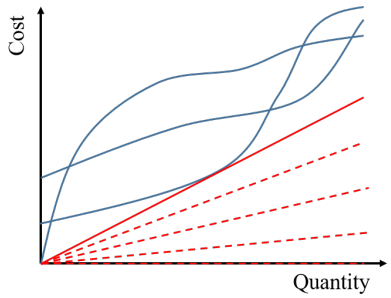


Figure 4.1: An illustration of the set Λ for an example with 3 non-convex cost functions. The three blue curves are the cost functions. The (dashed and solid) red lines lie below all the cost functions and their slopes are in Λ. The (slope of the) solid red line corresponds to the largest element of Λ.

all the solutions, i.e.,

$$\min_{\boldsymbol{q},\lambda,\boldsymbol{u}} \quad \sum_{i=1}^{n} u_i \tag{4.7a}$$

$$\text{s.t.} \quad (\boldsymbol{q},\lambda,\boldsymbol{u}) \in \underset{\boldsymbol{q},\lambda,\boldsymbol{u}}{\arg\min}\,(4.6a) \tag{4.7b}$$

$$\text{s.t. } (4.6b), (4.6c), (4.6d) \tag{4.7c}$$

where $\boldsymbol{q}$ and $\boldsymbol{u}$ denote $(q_1,\ldots,q_n)$ and $(u_1,\ldots,u_n)$, respectively.

Let us define Λ as the set of all λ's for which the linear price λq lies below all the cost functions, i.e.,

$$\Lambda = \{\lambda \geq 0 \mid \lambda q \leq c_i(q),\ \forall q, \forall i\}\,. \tag{4.8}$$

Figure 4.1 illustrates this set for an example with three non-convex costs.

The solutions to problems (4.6) and (4.7) can be found in closed-form, and the following summarizes the results.

Proposition 18. *The set of optimal solutions of problem* (4.6) *is given by*

$$\begin{cases} q_i^* = q_i^0,\ \forall i \\ \lambda^* \in \Lambda \\ u_i^* = c_i(q_i^*) - \lambda^* q_i^*,\ \forall i \end{cases}.$$

Proposition 19. *Problem* (4.7) *has a unique optimal solution as*

$$\begin{cases} q_i^* = q_i^0,\ \forall i \\ \lambda^* = \max\ \Lambda \\ u_i^* = c_i(q_i^*) - \lambda^* q_i^*,\ \forall i \end{cases}.$$

See the appendix for proofs.

Note that there were two potential alternatives to the two-stage optimization in (4.7) for picking a minimum-uplift solution. One may have attempted to enforce uniformity as a constraint. However, the problem with this is that imposing, for example, box constraints on u requires knowledge of reasonable upper-bounds on the uplifts, which may not be available; and on the other hand, insisting on exact uniformity makes the problem infeasible in most non-convex cases. The other alternative is to minimize a combination of the two objectives in (4.6) and (4.7). In this case, the weighted objective becomes $\sum_{i=1}^{n}(\lambda q_i + \gamma u_i)$ for some appropriate constant γ, and it is not hard to show that the solution will be the same as that of the proposed two-stage optimization.

4.3.2 An Efficient Approximation Algorithm

The optimization problem (4.5) defines a pricing rule that satisfies the desired properties in any non-convex market. For specific classes of cost functions, similar to the existing approaches, one may be able to solve this optimization problem using off-the-shelf solvers. For generic non-convex cost functions, however, there is no existing algorithm that can solve the optimization problem (4.5) to optimality. Furthermore, even finding an approximate solution, e.g., by discretizating the variables, requires a brute-force search, which quickly becomes intractable. In this section, we design a computationally efficient algorithm for solving the problem (4.5) approximately, based on decomposing it into smaller pieces, which works for general non-convex cost functions. This approximation algorithm can also be used to provide tractable calculations of some of the other non-convex pricing rules such as IP pricing.

Before going through the details of the algorithm, let us define the notion of an approximate solution to (4.5), which we consider. One could define an approximate solution as a value that is close enough, in a certain sense, to the optimal solution $(q_1^*, \ldots, q_n^*, \alpha^*, \beta_1^*, \ldots, \beta_n^*)$. However, no matter how close that approximation is to the optimal solution, that per se does not guarantee anything about the properties that the scheme will satisfy. Instead, we define an approximate solution to (4.5) as a set of quantities $q_1, \ldots, q_n$ and price parameters $\alpha, \beta_1, \ldots, \beta_n$ for which the Market Clearing condition holds exactly, the Revenue Adequacy and Competitive Equilibrium conditions are relaxed by an ϵ, and the total payment is at most $n\epsilon$ away from the optimal. More formally, it is defined as follows.

Definition 1. *$(q_1, \ldots, q_n, \alpha, \beta_1, \ldots, \beta_n)$ is called an **ϵ-approximate solution** to*

(4.5) *if it satisfies*

$$\sum_{i=1}^{n} q_i = d, \qquad \textit{(Market Clearing)}$$

$$p(q_i;\alpha,\beta_i) - c_i(q_i) + \epsilon \geq 0, \quad i = 1,\ldots,n, \qquad \textit{(ϵ-Revenue Adequacy)}$$

$$p(q_i;\alpha,\beta_i) - c_i(q_i) + \epsilon \geq p(q_i';\alpha,\beta_i) - c_i(q_i'), \qquad \forall q_i' \neq q_i, \quad i = 1,\ldots,n,$$

$$\textit{(ϵ-Competitive Equilibrium)}$$

and

$$\sum_{i=1}^{n} p(q_i;\alpha,\beta_i) \leq p^* + n\epsilon. \qquad \textit{(ϵ-Economic Efficiency)}$$

Given this notion of an approximate solution, we can move towards designing the algorithm. The optimization problem (4.5) looks highly coupled, at first, since the constraints share a lot of common variables. However, one can see that, for a fixed value of α, the objective becomes additively separable in (q_i, β_i). Furthermore (again for fixed α), constraints (4.5c),(4.5d) involve only the i-th variables (q_i, β_i) for each i. Although the Market Clearing condition still couples the variables together, this observation allows us to reformulate (4.5) as

$$p^* = \min_{\substack{q_1,\ldots,q_n \\ \alpha\in\mathcal{A}}} \quad \sum_{i=1}^{n} g_i(q_i;\alpha) \tag{4.9a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d, \tag{4.9b}$$

where

$$g_i(q;\alpha) = \min_{\beta_i\in\mathcal{B}} \quad p(q;\alpha,\beta_i) \tag{4.10a}$$

$$\text{s.t.} \quad p(q;\alpha,\beta_i) - c_i(q) \geq 0, \tag{4.10b}$$

$$p(q;\alpha,\beta_i) - c_i(q) \geq p(q';\alpha,\beta_i) - c_i(q'), \quad \forall q' \neq q, \tag{4.10c}$$

for all $i = 1,\ldots,n$.

Therefore, for any fixed value of α and q_i, the optimization over β_i can be done individually, as in (4.10). What remains to address, however, is the coupling of the variables as a result of the Market Clearing constraint. One naive approach would be to simply try all possible choices of $(q_1,\ldots,q_n)$ and pick the one that has the minimum objective value. This is very inefficient. Instead, we take a dynamic programming approach, and group pairs of variables together, defining a new variable

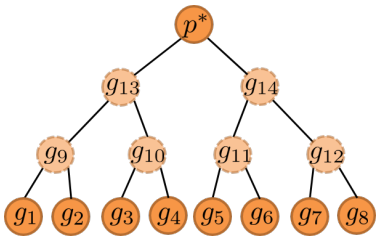


Figure 4.2: An example of the binary tree defined by Algorithm 1 for $n = 8$. The faded circles correspond to the added dummy nodes.

as their *parent*. We then group the parents together, and continue this process until we reach the *root*, i.e., where there is only one node. During this procedure, at each new node i, we need to solve the following (small) problem

$$\begin{aligned} g_i(q;\alpha) = \min_{q_j,q_k} \quad & g_j(q_j;\alpha) + g_k(q_k;\alpha) \\ \text{s.t.} \quad & q_j + q_k = q, \end{aligned} \tag{4.11}$$

for every q, where j and k are the *children* of i. At the root of the tree, we will be able to compute $g_{\text{root}}(d;\alpha)$. Figure 4.2 shows an example of the created binary tree for this procedure for $n = 8$. This procedure can be repeated for different values of α, and the optimal value p^* can be computed as $\min_\alpha g_{\text{root}}(d;\alpha)$.

The problem with recursion (4.11) is that it requires an infinite-dimensional computation at every step, since the values of $g_i(q;\alpha)$ need to be computed for every q. To get around this issue, we note that the variables q_i live in the bounded set $[0, d]$, and hence can be discretized to lie in a finite set Q, such that every possible q_i is at most $\delta(\epsilon)$ away from some point in Q. Similarly, if the α and β_i's are continuous variables, we can discretize the bounded sets $\mathcal{A}$ and $\mathcal{B}$ into some finite sets $\mathcal{A}'$ and $\mathcal{B}'$, such that every point in $\mathcal{A}$ (or $\mathcal{B}$) is at most $\delta(\epsilon)$ away, in infinity-norm sense, from some point in $\mathcal{A}'$ (or $\mathcal{B}'$). See the appendix for details.

For finding an ϵ-approximate solution, (4.10) is relaxed to

$$g_i(q;\alpha) = \min_{\beta_i \in \mathcal{B}'} \quad p(q;\alpha,\beta_i) \tag{4.12a}$$

$$\text{s.t.} \quad p(q;\alpha,\beta_i) - c_i(q) + \epsilon \geq 0, \tag{4.12b}$$

$$p(q;\alpha,\beta_i) - c_i(q) + \epsilon \geq p(q';\alpha,\beta_i) - c_i(q'), \quad \forall q' \neq q, \tag{4.12c}$$

for all $i = 1, \ldots, n$, and (4.11) remains the same, except the variables (q_j, q_k) take

Algorithm 1 Find an ϵ-approximate solution to the optimal pricing problem (4.5)

Input: n, $c_1(.), \ldots, c_n(.)$, $p(.;.)$, ϵ
for α in $\mathcal{A}'$ **do**
    $S = 1 : n$
    for i in S **do** ▷ for the leaves
        compute $g_i(q;\alpha)$ **for all** q in Q, using (4.12)
    end for
    while $|S| > 2$ **do** ▷ while not reached the root
        $S_{\text{new}} = S(\text{end}) + 1 \ : \ S(\text{end}) + \left\lceil \frac{|S|}{2} \right\rceil$
        for i in S_{new} **do** ▷ for the intermediate nodes
            $[j, k]$ = indices of children of i
            if $k = \emptyset$ **then** $g_i(.;\alpha) = g_j(.;\alpha)$
            else, compute $g_i(q;\alpha)$ **for all** q in Q, using (4.13) ▷ it has two children
            end if
        end for
        $S = S_{\text{new}}$
    end while
    $[j, k] = S$
    compute $g_{\text{root}}(d;\alpha)$, using (4.13) ▷ at the root
end for
$\alpha^* = \underset{\alpha \in \mathcal{A}'}{\arg\min} \quad g_{\text{root}}(d;\alpha)$
$q^*_{\text{root}} = d$
for i = root $: -1 : n + 1$ **do**
    $[q^*_j, q^*_k] = x_i(q^*_i;\alpha^*)$, where $[j, k]$ = indices of children of i
end for
for $i = n : -1 : 1$ **do**
    $\beta^*_i = b_i(q^*_i;\alpha^*)$
end for
return $(q^*_1, \ldots, q^*_n, \alpha^*, \beta^*_1, \ldots, \beta^*_n)$

values in Q, i.e.,

$$\begin{aligned} g_i(q;\alpha) = \min_{q_j, q_k \in Q} \quad & g_j(q_j;\alpha) + g_k(q_k;\alpha) \\ \text{s.t.} \quad & q_j + q_k = q, \end{aligned} \tag{4.13}$$

for all $i > n$. We denote the optimizer of (4.12) by $b_i(q;\alpha)$, and the optimizer of (4.13), which is a pair of quantities (q_j, q_k), by $x_i(q;\alpha)$. The full procedure is summarized in pseudocode in Algorithm 1.

While not immediately clear, the proposed approximation algorithm can be shown to run in time that is polynomial in both n and $1/\epsilon$ (in fact, linear in n). Further, the solution it provides is ϵ-accurate under a mild smoothness assumption on the cost and price functions, which holds true for almost any function considered in the literature. These two results are summarized in the following theorem, which is proven in the appendix.

Theorem 20. *Consider $c_i(.)$ and $p(.;.)$ that have at most a finite number of discontinuities and are Lipschitz on each continuous piece of their domain. Algorithm 1 finds an ϵ-approximate solution to the optimal pricing problem* (4.5) *with running time $O\left(n(1/\epsilon)^{l_1+l_2+2}\right)$, where n is the number of suppliers, and l_1 and l_2 are the number of shared and individual parameters in the price, respectively.*

It is worth emphasizing that while there are $l_1 + nl_2$ variables in the price functions in total, parameters l_1 and l_2 do not scale with n, and are typically very small constants. For example, for the so-called linear-plus-uplift price functions $l_1 = l_2 = 1$. Therefore, the algorithm is very efficient.

We should also remark that if one requires the total payment in Definition 1 to be at most ϵ (rather than $n\epsilon$) away from the optimal p^*, the running time of our algorithm will still be polynomial in both n and $1/\epsilon$, i.e., $O\left(n^3(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$. See the appendix for details.

4.4 Equilibrium-Constrained Pricing for Networked Markets

We now consider the more general problem of finding an efficient pricing scheme in a networked market. The networked market we consider has n suppliers, located at the nodes (vertices) $V = \{1, \ldots, n\}$ of a network, and connected through lines (edges) E, where, without loss of generality, the edges are defined to be from the smaller node to the larger node (i.e., $\forall (i,j) \in E,\ i < j$). The i-th supplier has a cost function $c_i(q_i)$ for producing quantity q_i, which may be non-convex, as before, and

there is an inelastic demand d_i at each node i. The lines connecting the nodes can possibly have certain capacities for the flows they can carry. We denote the flow of any line $e = (i, j)$, from i to j, by f_e, and its limits (capacity) by $\underline{f_e}$ and $\overline{f_e}$ (the flow from j to i is $-f_e$).

Note that if there are multiple suppliers co-located in a market, we can simply assign them each their own vertex, and connect them through paths with infinite capacities. In other words, a node with multiple suppliers can be simply replaced with a "line graph" composed of those suppliers and infinite-capacity edges.

4.4.1 Pricing Formulation

A key benefit of EC pricing is the ease of generalization to the networked setting. There are no current pricing rules that can be readily applied to the networked case. In this setting, our Equilibrium-Constrained pricing can be formulated as the following optimization problem.

Networked Equilibrium-Constrained (EC) Pricing:

$$p^* = \min_{\substack{q_1,\ldots,q_n \\ \{f_e\}_{e\in E} \\ p_1,\ldots,p_n\in\mathcal{P}}} \quad \sum_{i=1}^{n} p_i(q_i) \tag{4.14a}$$

$$\text{s.t.} \quad q_i - d_i = \sum_{\substack{j \\ (i,j)\in E}} f_{(i,j)} - \sum_{\substack{j \\ (j,i)\in E}} f_{(j,i)}, \quad i = 1,\ldots,n \tag{4.14b}$$

$$\underline{f_e} \le f_e \le \overline{f_e}, \quad e \in E \tag{4.14c}$$

$$p_i(q_i) - c_i(q_i) \ge 0, \quad i = 1,\ldots,n \tag{4.14d}$$

$$p_i(q_i) - c_i(q_i) \ge \max_{q_i' \ne q_i} p_i(q_i') - c_i(q_i'), \quad i = 1,\ldots,n \tag{4.14e}$$

The objective is the total payment, as discussed before, and the optimization is over quantities q_i, line flows f_e, and price functions $p_i \in \mathcal{P}$. Constraint (4.14b) is the Market Clearing condition (or Flow Conservation) for each individual node, i.e., the net production at each node should be equal to its outgoing flow. Constraint (4.14c) enforces the line limits (Capacity Constraints). Constraints (4.14d) and (4.14e) are Revenue Adequacy and Competitive Equilibrium, respectively, as before. The key difference between the networked setting and the single-market one is that here, the Market Clearing condition is spread across the network, and we have to solve the problem for the flows as well.

Remark. *When the capacity constraints* (4.14c) *are relaxed (*$\underline{f_e} = -\infty, \overline{f_e} = \infty, \forall e \in E$*), the networked problem reduces to the single-market one. In this case, the solution to the optimization problem* (4.14) *reduces to that of* (4.1)*. That is because the only constraint involving the flows would be* (4.14b)*, and we can always finds flows that satisfy it, as long as* $\sum_{i=1}^{n} q_i - \sum_{i=1}^{n} d_i = 0$*, which is the conventional Market Clearing condition.*

Assuming a parametric form $p_i(q_i) \coloneqq p(q_i; \alpha, \beta_i)$ for $\mathcal{P}$, with shared parameters α and individual parameters β_i as before, the proposed pricing can be expressed as follows.

Parameterized Networked Equilibrium-Constrained (EC) Pricing:

$$p^* = \min_{\substack{q_1,\dots,q_n \\ \{f_e\}_{e\in E} \\ \alpha\in\mathcal{A} \\ \beta_1,\dots,\beta_n\in\mathcal{B}}} \quad \sum_{i=1}^{n} p(q_i; \alpha, \beta_i) \tag{4.15a}$$

$$\text{s.t.} \quad q_i - d_i = \sum_{\substack{j \\ (i,j)\in E}} f_{(i,j)} - \sum_{\substack{j \\ (j,i)\in E}} f_{(j,i)}, \quad i = 1, \dots, n \tag{4.15b}$$

$$\underline{f_e} \le f_e \le \overline{f_e}, \quad e \in E \tag{4.15c}$$

$$p(q_i; \alpha, \beta_i) - c_i(q_i) \ge 0, \quad i = 1, \dots, n \tag{4.15d}$$

$$p(q_i; \alpha, \beta_i) - c_i(q_i) \ge \max_{q_i' \ne q_i} \; p(q_i'; \alpha, \beta_i) - c_i(q_i'), \quad i = 1, \dots, n \tag{4.15e}$$

4.4.2 An Efficient Approximation Algorithm

For certain classes of non-convexities, the optimization problem (4.15) can still be solved using off-the-shelf solvers, similar to those used in the other methods for the no-network case. However, those algorithms cannot handle more general classes of non-convexities. In this section, we develop a computationally efficient approximation algorithm for general non-convex costs, for a special class of networks.

A special yet important class of networks are *acyclic* networks, which are a typical topology in many markets, including *electricity distribution networks*. Acyclic networks have a *tree* topology (they do not have cycles), which allows us to devise an efficient algorithm for them. In the remainder of this section, we limit our attention to these networks. The main ideas extend directly to more general networks, as

long as there are not "too many cycles" in the network in some sense (i.e., bounded tree-width networks). We have focused on the acyclic case due to space constraints.

Without loss of generality, let us denote the first node as the *root* of the tree, and nodes with only one neighbor as the *leaves*. Every node (except the root) has a unique *parent*, defined as the first node on the unique path connecting it to the root node. The set of nodes that have a given node i as their parent is said to be node i's *children*. It can be shown that any tree with arbitrary degree can be transformed into a *binary tree*, i.e., a tree where each node has a unique parent and at most 2 children, with $O(n)$ nodes (see the appendix). Thus, we can focus on binary trees.

For a node i, let $\mathrm{ch}_1(i), \mathrm{ch}_2(i)$ denote its children ($\mathrm{ch}_1(i) = \emptyset$ and/or $\mathrm{ch}_2(i) = \emptyset$ when i has less than two children). The problem can then be written as

$$p^* = \min_{\substack{q_1,\dots,q_n \\ f_1,\dots,f_n \\ \alpha\in\mathcal{A} \\ \beta_1,\dots,\beta_n\in\mathcal{B}}} \quad \sum_{i=1}^{n} p(q_i;\alpha,\beta_i) \tag{4.16a}$$

$$\text{s.t.} \quad q_i - d_i = f_{\mathrm{ch}_1(i)} + f_{\mathrm{ch}_2(i)} - f_i, \quad i = 1,\dots,n \tag{4.16b}$$

$$\underline{f_i} \le f_i \le \overline{f_i}, \quad i = 1,\dots,n \tag{4.16c}$$

$$p(q_i;\alpha,\beta_i) - c_i(q_i) \ge 0, \quad i = 1,\dots,n \tag{4.16d}$$

$$p(q_i;\alpha,\beta_i) - c_i(q_i) \ge \max_{q_i'\neq q_i} \; p(q_i';\alpha,\beta_i) - c_i(q_i'), \quad i = 1,\dots,n \tag{4.16e}$$

where f_i represents the incoming flow to each node i from its parent, and $\underline{f_{\mathrm{root}}} = \overline{f_{\mathrm{root}}} = 0$.

Similarly as in the single-market case, we define an ϵ-approximate solution to this problem.

Definition 2. $(q_1,\dots,q_n,f_1,\dots,f_n,\alpha,\beta_1,\dots,\beta_n)$ *is called an* ϵ-approximate so-

lution *to* (4.16) *if it satisfies*

$$
\begin{aligned}
&|q_i - d_i - f_{\mathrm{ch}_1(i)} - f_{\mathrm{ch}_2(i)} + f_i| \le \epsilon, \quad i = 1, \ldots, n, && \textit{(ϵ-Load Balancing)}\\
&\underline{f_i} \le f_i \le \overline{f_i}, \quad i = 1, \ldots, n, && \textit{(Flow Limit)}\\
&p(q_i; \alpha, \beta_i) - c_i(q_i) + \epsilon \ge 0, \quad i = 1, \ldots, n, && \textit{(ϵ-Revenue Adequacy)}\\
&p(q_i; \alpha, \beta_i) - c_i(q_i) + \epsilon \ge p(q_i'; \alpha, \beta_i) - c_i(q_i'), \qquad \forall q_i' \ne q_i, \quad i = 1, \ldots, n, && \\
& && \textit{(ϵ-Competitive Equilibrium)}\\
&\sum_{i=1}^{n} p(q_i; \alpha, \beta_i) \le p^* + n\epsilon. && \textit{(ϵ-Economic Efficiency)}
\end{aligned}
$$

The main difference from the definition in the single-market case is that the Market Clearing condition has been replaced with ϵ-Load Balancing and exact Flow Limit conditions here.

Note that the minimization over the variables β_i in problem (4.16) can be done "internally," and the problem can be re-expressed as

$$
\begin{aligned}
p^* = \min_{\substack{q_1,\ldots,q_n \\ f_1,\ldots,f_n \\ \alpha \in A}} \quad & \sum_{i=1}^{n} g_i(q_i; \alpha) && \text{(4.17a)}\\
\text{s.t.} \quad & q_i - d_i = f_{\mathrm{ch}_1(i)} + f_{\mathrm{ch}_2(i)} - f_i, \quad i = 1, \ldots, n && \text{(4.17b)}\\
& \underline{f_i} \le f_i \le \overline{f_i}, \quad i = 1, \ldots, n && \text{(4.17c)}
\end{aligned}
$$

where

$$
\begin{aligned}
g_i(q; \alpha) = \min_{\beta_i \in \mathcal{B}} \quad & p(q; \alpha, \beta_i) && \text{(4.18a)}\\
\text{s.t.} \quad & p(q; \alpha, \beta_i) - c_i(q) \ge 0, && \text{(4.18b)}\\
& p(q; \alpha, \beta_i) - c_i(q) \ge p(q'; \alpha, \beta_i) - c_i(q'), \quad \forall q' \ne q, && \text{(4.18c)}
\end{aligned}
$$

for all $i = 1, \ldots, n$.

The key insight is that the tree structure of the constraints (4.17b) allows us to write the optimization problem in a recursive form as follows:

$$
p^* = \min_{\alpha} \; h_{\mathrm{root}}(0; \alpha) \tag{4.19}
$$

Algorithm 2 Find an ϵ-approximate solution to the optimal networked pricing problem (4.16)

Input: G=(V,E), $c_1(.), \ldots, c_n(.), p(.;.), \epsilon$
for α in $\mathcal{A}'$ **do**
  for all nodes i **do**
    compute $g_i(q_i;\alpha)$ **for all** q_i in Q_i, using (4.22)
  end for
  for all nodes $i \neq$root (in bottom-up order) **do**
    compute $h_i(f;\alpha)$ **for all** f in F_i, using (4.21)
  end for
  compute $h_{\text{root}}(0;\alpha)$, using (4.21)
end for
$\alpha^* = \underset{\alpha\in\mathcal{A}'}{\arg\min} \quad h_{\text{root}}(0;\alpha)$
$f^*_{\text{root}} = 0$
for all nodes i (in top-down order) **do**
  $[q_i^*, f^*_{\text{ch}_1(i)}, f^*_{\text{ch}_2(i)}] = y_i(f_i^*;\alpha^*)$
  $\beta_i^* = b_i(q_i^*;\alpha^*)$
end for
return $(q_1^*, \ldots, q_n^*, f_1^*, \ldots, f_n^*, \alpha^*, \beta_1^*, \ldots, \beta_n^*)$

where

$$h_i(f_i;\alpha) = \min_{q_i, f_{\text{ch}_1(i)}, f_{\text{ch}_2(i)}} \quad g_i(q_i;\alpha) + h_{\text{ch}_1(i)}(f_{\text{ch}_1(i)};\alpha) + h_{\text{ch}_2(i)}(f_{\text{ch}_2(i)};\alpha) \tag{4.20a}$$

$$\text{s.t.} \quad q_i - d_i = f_{\text{ch}_1(i)} + f_{\text{ch}_2(i)} - f_i \tag{4.20b}$$

$$\underline{f_{\text{ch}_1(i)}} \leq f_{\text{ch}_1(i)} \leq \overline{f_{\text{ch}_1(i)}} \tag{4.20c}$$

$$\underline{f_{\text{ch}_2(i)}} \leq f_{\text{ch}_2(i)} \leq \overline{f_{\text{ch}_2(i)}} \tag{4.20d}$$

for all $i = 1, \ldots, n$.

Now, this recursive form is amenable to dynamic programming. However, since the variables are continuous, each step still requires an infinite-dimensional search. In order to tackle this issue, we can discretize the variables and solve the following approximate versions.

$$h_i(f_i;\alpha) = \min_{\substack{q_i\in Q_i \\ f_{\text{ch}_1(i)}\in F_{\text{ch}_1(i)} \\ f_{\text{ch}_2(i)}\in F_{\text{ch}_2(i)}}} \quad g_i(q_i;\alpha) + h_{\text{ch}_1(i)}(f_{\text{ch}_1(i)};\alpha) + h_{\text{ch}_2(i)}(f_{\text{ch}_2(i)};\alpha) \tag{4.21a}$$

$$\text{s.t.} \quad |q_i - d_i - f_{\text{ch}_1(i)} - f_{\text{ch}_2(i)} + f_i| \leq \epsilon \tag{4.21b}$$

for all $i = 1, \ldots, n$, where $Q_1, \ldots, Q_n$ and $F_1, \ldots, F_n$ are properly-defined discrete sets (see the appendix for details). We denote the optimizer (triple) of (4.21) by

$y_i(f_i;\alpha)$.

$$g_i(q;\alpha) = \min_{\beta_i \in \mathcal{B}'} \quad p(q;\alpha,\beta_i) \tag{4.22a}$$

$$\text{s.t.} \quad p(q;\alpha,\beta_i) - c_i(q) + \epsilon \geq 0, \tag{4.22b}$$

$$p(q;\alpha,\beta_i) - c_i(q) + \epsilon \geq p(q';\alpha,\beta_i) - c_i(q'), \quad \forall q' \neq q, \tag{4.22c}$$

for all $i = 1, \ldots, n$. The optimizer of (4.22) is denoted by $b_i(q;\alpha)$.

The steps of the procedure are summarized in pseudocode in Algorithm 2, and the following result summarizes the theoretical guarantee of the algorithm.

Theorem 21. *Consider $c_i(.)$ and $p(.;.)$ that have at most a finite number of discontinuities and are Lipschitz on each continuous piece of their domain. Algorithm 2 finds an ϵ-approximate solution to the optimal networked pricing problem* (4.16)*, with running time $O\left(n(1/\epsilon)^{l_1+\max\{l_2,1\}+2}\right)$, where n is the number of suppliers, and l_1 and l_2 are the number of shared and individual parameters in the price, respectively.*

It is worth mentioning that the network algorithm developed in this section suggests another way of solving the no-network case as well, by replacing the single market with a line graph with infinite capacities. This algorithm will in turn have time complexity $O\left(n(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$, which is the same as that of the one developed in Section 4.3.2.

4.5 Existing Pricing Schemes

In this section, we review the existing pricing schemes in the literature and summarize their properties. No prior pricing rule for general non-convex markets satisfies all the properties discussed in Section 4.2.2. However, it is possible to achieve all the properties in the case when the cost functions are convex via a classical approach: *shadow pricing*. We first briefly illustrate how shadow pricing works for the convex case, and then survey some prominent approaches in the literature that seek to extend the properties of shadow pricing to the non-convex case, contrasting them with the EC scheme.

4.5.1 Pricing in Convex Markets

When the cost functions $c_i(.)$ are convex, a simple and uniform pricing rule, often referred to as *shadow pricing* or *marginal-cost pricing* [201, 32], can achieve all the above-mentioned properties. The pricing scheme works as follows. The operator

first solves the convex program

$$\min_{q_1,\ldots,q_n} \quad \sum_{i=1}^{n} c_i(q_i) \tag{4.23a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d \qquad (\lambda) \tag{4.23b}$$

where λ is the dual variable corresponding to the load-balance constraint. Let $q_1^*, \ldots, q_n^*$ and λ^* denote an optimal primal-dual pair of this problem (if there are multiple dual solutions, take λ^* to be the smallest). A payment function of the form

$$p_i(q_i) = \lambda^* q_i \quad i = 1, \ldots, n \tag{4.24}$$

satisfies all the properties outlined in Section 4.2.2, and it is relatively straightforward to see that.

For simplicity, assume that $c_i(.)$ are differentiable. The optimal solution of (4.23) satisfies the following (KKT) conditions (which does not require convexity):

$$\begin{cases} \sum_{i=1}^{n} q_i^* = d \\ \frac{dc_i}{dq_i}(q_i^*) = \lambda^*, \quad i = 1, \ldots, n \end{cases}$$

Next, note that supplier i's profit-maximization problem is

$$\max_{q_i} \quad \lambda^* q_i - c_i(q_i).$$

Since $c_i(.)$ is convex, the objective is concave, and any point at which the derivative is zero is a global maximizer. In particular, the derivative at q_i^* is zero, because of the KKT conditions, and therefore that is a solution to the supplier i's profit-maximization problem. As a result, the scheme supports a competitive equilibrium that clears the market and minimizes the production cost, while using a price form that is simple and uniform. Figure 4.3 illustrates the optimal quantities and the price function for an example with three suppliers.

Note that the total payment of this scheme is $\sum_{i=1}^{n} p_i(q_i^*) = \lambda^* d$, which can be generally higher than $\sum_{i=1}^{n} c_i(q_i^*)$. One can always opt for a *non-uniform* affine price function as $p_i(q_i) = \lambda^* q_i + b_i$, with $b_i = c_i(q_i^*) - \lambda^* q_i^*$, which has lower payments, and makes $\sum_{i=1}^{n} p_i(q_i^*)$ exactly equal to $\sum_{i=1}^{n} c_i(q_i^*)$. However, if one requires a *uniform and linear* price function, it can be shown that $p_i(q_i) = \lambda^* q_i$ has the *lowest total payment* among all such functions.

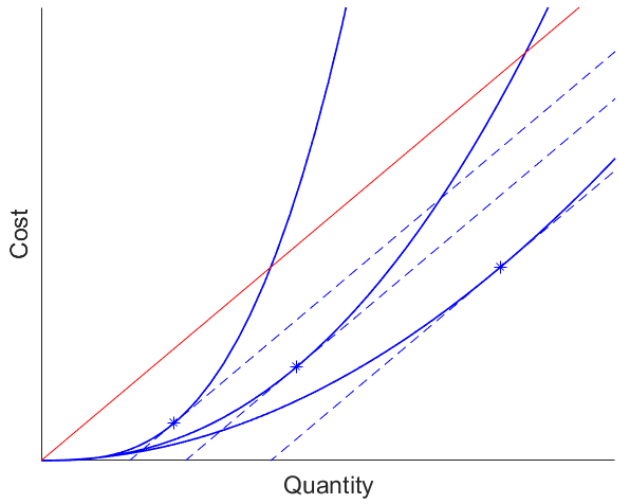


Figure 4.3: An illustration of shadow pricing for the case of 3 convex cost functions. The points indicated by $*$ show the optimal quantities. The 3 functions have the same derivative at their optimal quantities, and the tangent line lies below the function (because of convexity). The red (solid) line that passes through the origin is the uniform price function, which is parallel to the three lines.

4.5.2 Pricing in Non-Convex Markets

If the cost functions are non-convex, the approach of shadow pricing, described above, fails. This is because the net profit of each supplier is no longer a concave function, and its stationary points do not necessarily correspond to the maximum. In other words, there may not be a subderivative at q_i^* supporting the cost function $c_i(.)$.

There have been several schemes proposed in the literature that attempt to address this issue and design pricing rules that satisfy the properties discussed above in the context of non-convex cost functions. We review the most promising ones here. Some of the schemes maintain a uniform pricing rule with additional discriminatory side-payments called "uplifts" for incentivizing the suppliers to follow the dispatch, while others raise the uniform price so that it is revenue-adequate. A summary of the pricing schemes, along with their properties, is provided in Table 4.1.

Integer Programming (IP)

[159] proposed a pricing scheme for non-convex cost functions that are in the form of a fixed (start-up) cost plus a linear marginal cost, sometimes referred to as "IP pricing." This scheme uses uniform marginal pricing for the commodity and discriminatory

Table 4.1: Summary of common pricing schemes and their properties.

Scheme \ Property	Price form $p_i(q_i) =$	Proposed for $c_i(q_i) =$	Market Clearing	Revenue Adequate	Supports Competitive Equilibrium	Economically Efficient
Shadow Pricing	λq_i	Convex	✓	✓	✓	✓
IP	$\lambda q_i + u_i \mathbb{1}_{q_i>0}$	Startup+linear	✓	✓	✓	✓
MU/CH	$\lambda q_i + u_i \mathbb{1}_{q_i=q_i^*}$	Startup+convex	✓	✓	✓	×
SLR	λq_i	Startup+linear	✓	✓	×	×
PD	λq_i	Startup+linear	✓	✓	×	×
EC (proposed)	User-specified	General	✓	✓	✓	✓

Notes. IP: Integer Programming. MU: Minimum Uplift. CH: Convex Hull. SLR: Semi-Lagrangean Relaxation. PD: Primal-Dual. EC: Equilibrium-Constrained. The results in this table assume solving the formulation for each scheme exactly. However, in practice, these schemes rely on numerical solvers for their problems, and if the problem is non-convex, there is no guarantee of maintaining these properties in general. In particular, the IP scheme requires a non-convex solver. The MU/CH, SLR, and PD schemes, for the cost functions that they are proposed for (i.e., startup+convex or startup+linear), require only convex solvers and therefore satisfy the checked properties exactly. The EC scheme is accompanied by an efficient algorithm for solving the non-convex problem for general cost functions, which satisfies the exact Market Clearing property and the ϵ-approximate versions of the other three properties (see Section 4.3.2).

pricing for the integral activity of the suppliers. It is based on (i) formulating an optimization similar to (4.23), as a mixed integer linear program (MILP) and solving it for optimal allocations, (ii) reformulating the original MILP as an LP by replacing the integral constraints with forcing commitment choices equal to their optimal values, and (iii) solving the LP problem and using the dual variable λ of Market Clearing constraint as the uniform price and the dual variables $\{u_i^*\}$ of the forced equality constraints as discriminatory uplifts: $p_i(q_i) = \lambda^* q_i + u_i^* \mathbb{1}\{q_i > 0\}$.

IP pricing uses a uniform price plus a discriminatory uplift to clear the market efficiently such that every supplier's net profit is zero. As a result, both total payments and total production costs are minimized at the same time. [159] show that the optimal solutions generated by IP pricing are optimal to the decentralized profit maximization problems for every supplier, and thus they support a competitive equilibrium. However, IP pricing assumes knowledge of the optimal solutions to the unit commitment problem and thus is not intended as a practical approach to find the optimal allocation. [103] point out that uniform price generated under IP pricing can be volatile (i.e., a small change in demand could lead to a big change in the uniform price) and uplifts could be generally very large.

Minimum Uplift (MU) / Convex Hull (CH)

To avoid the unwanted properties of IP pricing (i.e., volatility and instability), a pricing scheme, proposed in [103] for the (non-convex) class of startup-plus-convex cost functions, offers minimum uplifts that incentivize each supplier to follow the dispatch rather than maximize their own profits in the absence of uplifts. The scheme is based on solving the mixed-integer program minimizing the total production cost and minimizing total uplifts. Given a fixed uniform price λ, each supplier chooses between following the dispatch to receive the uplifts or not. The uplifts can be viewed as the extra potential profit that the suppliers can make by self-scheduling and maximizing their own profit. [81] refined the MU pricing and proposed the concept of Convex Hull pricing, which is based on (i) replacing the non-convex cost of the original program with its convex hull to formulate a new LP and (ii) solving the new LP and using the dual variable of the Market Clearing constraint as the marginal price and deriving the lost opportunity cost as the minimum uplifts to incentivize suppliers' compliance. The final payment $p_i(q_i, z_i)$ as a function of quantity q_i and commitment choice z_i is in the form of a uniform price λ^* and a discriminatory uplift u_i^* as $p_i(q_i) = \lambda^* q_i + u_i^* \mathbb{1}\left\{q_i = q_i^*\right\}$.

Even though MU/CH pricing minimizes total uplifts, the generated marginal price might end up being high, and the payments can be much higher than those of the other schemes. In general, the total payments under this scheme might end up being much higher than the total production costs, which defeats the purpose of minimizing the costs. Even for the class of startup-plus-linear cost functions, where IP pricing is optimal (the total payment is equal to the total production cost, and they are both minimal), MU pricing is not economically efficient, as it fails to minimize the payments.

On the computational side, although [107] propose a polynomially-solvable primal formulation for the Lagrangian dual problem by explicitly describing the convex hull for piecewise linear or quadratic cost functions, describing the convex hull of cost functions could be very challenging in general and thus makes the problem computationally intractable.

As an aside, MU and CH would not be equivalent if the Market Clearing constraint was an inequality. In that case, the side-payments in CH would be typically larger than those in MU, due to Product Revenue Shortfall [181].

Semi-Lagrangean Relaxation (SLR)

[10] introduced a semi-Lagrangean relaxation approach to find a uniform price that is revenue-adequate at the same solution for quantity and commitment choices as the original optimization problem, for cost functions of startup-plus-linear form. The scheme is based on formulating and solving the SLR of the mixed-integer program by semi-relaxing the Market Clearing constraint with standard Lagrange multiplier λ. The solution under SLR satisfies the constraints in the original MIP and makes the duality gap between MILP and SLR zero. Though the payment function $p_i(q_i) = \lambda^* q_i$ under SLR pricing is high enough to avoid negative profits for suppliers, it incentivizes the suppliers to deviate and operate at full capacity, and total payments usually end up being much higher than total costs of production.

Primal-Dual (PD)

Another revenue-adequate pricing scheme, proposed by [177], exploits a primal-dual approach to derive a uniform price to guarantee that dispatched suppliers are willing to remain in the market (revenue adequacy). The scheme works for cost functions with the form of start-up cost plus linear cost, and the prices have shown not to deviate much from that of [159]. The approach is based on (i) relaxing the integral constraint of the original MILP to formulate a primal LP problem, (ii) deriving the dual LP problem of the primal LP problem, (iii) formulating a new LP problem that seeks to minimize the duality gap between the primal and dual problems subject to both primal and dual constraints, and (iv) adding back the integral constraints as well as nonlinear constraints to ensure that no supplier incurs loss and solving the new problem for optimal solutions q_i^*, z_i^* and λ^*.

Though this scheme may be implemented using standard branch-and-cut solvers, it is computationally intractable in general. The prices $p_i(q_i) = \lambda^* q_i$ and profits produced under PD do not significantly deviate from dual prices if integral constraints are relaxed and thus are always bounded. However, as a revenue-adequate pricing scheme, PD fails to form a competitive equilibrium as suppliers are incentivized to operate at full capacity. In general, total payments are much higher than total production costs.

4.6 Experimental Results

In this section, we compare and contrast EC pricing with the existing approaches using numerical experiments on common case studies. Specifically, we compare

the payments and uplifts generated from different pricing schemes, including IP, CH, SLR, PD, and EC. Among all these schemes, only EC allows flexibility of the payment form. As a result, we further divide EC into one with a payment function in the form of linear marginal price plus uplifts and another pricing with a payment form of piecewise linear marginal prices plus uplifts. In practice, specific limits on the number of sections and the maximum slope among all sections can be used to further restrict EC. For convenience, we name these variations of EC in terms of number of piecewise sections of its payment form, e.g., EC2 refers to EC with a payment function in the form of 2 piecewise sections plus uplifts.

First, we apply all these pricing schemes to a *single* market example from [103], which is a modification of Scarf's example developed in [179]. Second, we adapt cost functions in the modified Scarf's example to be quadratic plus startup cost in order to further explore how these schemes generalize to different cost functions. Finally, we consider a further generalization to a simple 2-node networked market.

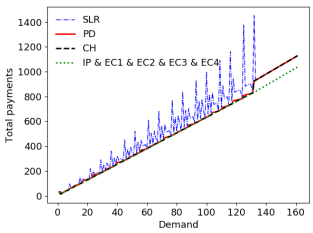
4.6.1 Case 1: Linear plus startup cost

Table 4.2: Summary of the production characteristics in the modified Scarf's example.

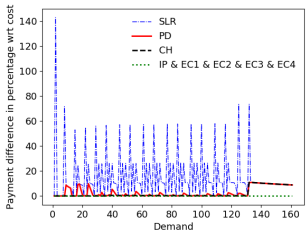
Type	Smokestack	High Tech	Med Tech
Capacity	16	7	6
Minimum output	0	0	2
Startup cost	53	30	0
Marginal cost	3	2	7
Quantity	6	5	5

We consider a modified Scarf's example, as proposed in [103]. The parameters are listed in Table 4.2. We assume that demand is inelastic with a maximum capacity of 161 units. We restrict the payment function of EC1, EC2, EC3, and EC4 to respectively have one, two, three, and four sections and impose that the marginal price of any section cannot exceed the maximum marginal price for any supplier operating at full capacity. Figure 4.4a shows total payments for different demand levels, while Figure 4.4c shows the corresponding uplifts of the pricing schemes that apply, i.e., CH, EC1, EC2, EC3, and EC4. Payments of two revenue-adequate pricing schemes, including SLR and PD, are higher than total costs in general. IP, EC1, EC2, EC3, and EC4 achieve the minimum payments equal to total costs. CH achieves the minimum payments at low demand levels, and its total payments surpass total costs as demand gets high. As for uplifts, EC4 achieves the smallest among

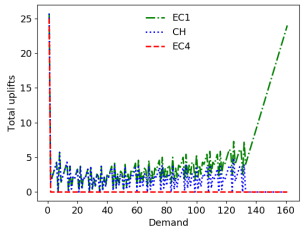
the five pricing schemes. Total uplifts of CH and EC1 are close to each other at a low demand level and that of EC1 increases significantly when demand approaches capacity. This is not surprising, as total payments of CH go above total costs at a high demand, making it possible for relatively smaller total uplifts. It is worth noting that startup prices and marginal prices for IP are volatile and unstable. Figure 4.4d and 4.4e demonstrate that the more complex we allow payment functions of the EC family, the smaller total uplifts we can achieve, which means more uniform prices are across suppliers. In practice, there is apparently a trade-off between complexity and uniformity of payment functions among the EC family, and this will be a design choice for the independent system operator (ISO). Overall, EC4 outperforms other pricing schemes in terms of total payments and total uplifts.



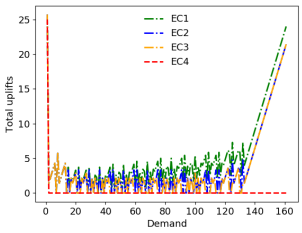
(a) Total payments as a function of demand.



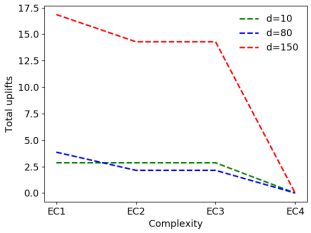
(b) Payment difference in percentage w.r.t cost as a function of demand.



(c) Total uplifts as a function of demand.



(d) Total uplifts as a function of demand.



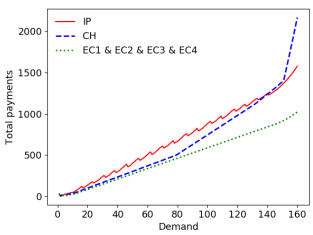
(e) Total uplifts for pricing schemes at different demand levels.

Figure 4.4: An example with cost functions of the form of linear plus startup cost.

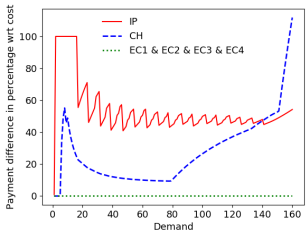
4.6.2 Case 2: Quadratic plus startup cost

Table 4.3: Summary of the new cost functions in the modified Scarf's example.

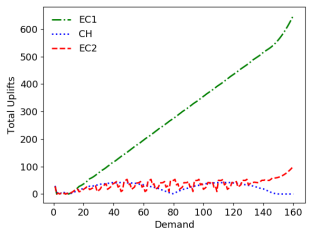
Type	Smokestack	High Tech	Med Tech
Cost function	$\frac{3}{16}q^2 + 53 * \mathbb{1}\{q > 0\}$	$\frac{2}{7}q^2 + 30 * \mathbb{1}\{q > 0\}$	$\frac{7}{6}q^2$



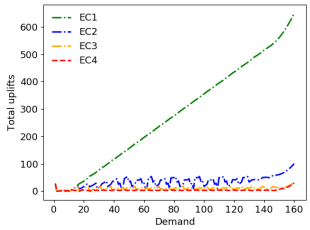
(a) Total payments as a function of demand.



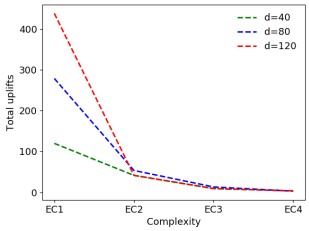
(b) Payment difference in percentage w.r.t cost as a function of demand.



(c) Total uplifts as a function of demand.



(d) Total uplifts as a function of demand.



(e) Total uplifts for pricing schemes at different demand levels.

Figure 4.5: An example with cost functions of the form of quadratic plus startup cost.

To further explore how these pricing schemes generalize to different cost functions, we modify the cost functions of the example above. Table 4.3 describes the new cost functions for each supplier. Since it is not clear how to generalize SLR and PD, we focus on a comparison among IP, CH, EC1, EC2, EC3, and EC4. We restrict the payment function of EC1, EC2, EC3, and EC4 to respectively have one, two, three, and four sections with the marginal price of any section bounded by the maximum of marginal price for any supplier operating at full capacity. As can be seen in Figure 4.5a, EC1 EC2, EC3, and EC4 achieve the possible minimum total payments equal to total costs. Total payments of IP and CH are both above total costs, and the gap between total payments and costs grows as demand increases. Observe that the demand here ranges from 1 to 160 because marginal price of CH increases dramatically at the capacity level, and the plot over the interval $(1, 160)$ would be a flat line if the whole range were covered. Figure 4.5c shows that total uplifts of EC1 are much larger than that of CH and EC2. At a low demand level, uplifts of EC1 and EC2 are close to each other. As demand increases, uplifts of EC2 are a little larger than those of CH, in order to maintain a smaller overall payment. There is a trade-off between minimizing total payments and minimizing total costs. Allowing the flexibility of payment function form enables EC2 to perform better than either CH or EC1 in terms of total payments and uplifts. Figure 4.5d and 4.5e

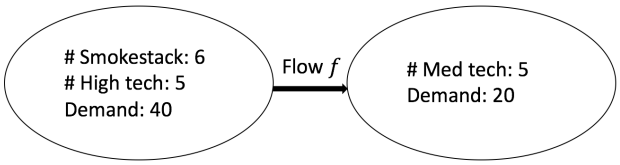
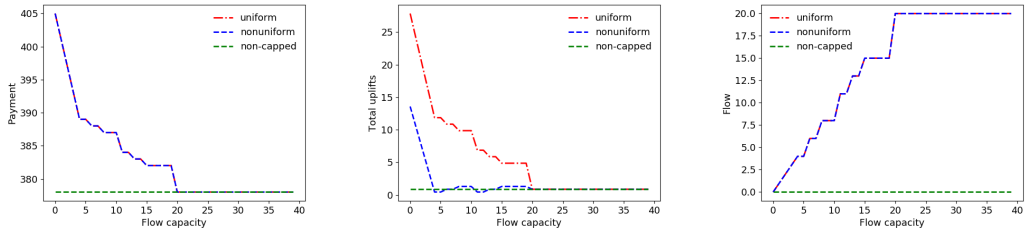


Figure 4.6: A schematic drawing for two connected markets with a constraint on flow capacity.

show a relationship between complexity of payment function form and magnitude of total uplifts among the EC family pricing schemes. As in the case of cost function being start-up plus linear cost, it is not surprising to see that more complex payment functions tend to allow smaller total uplifts, i.e., more uniform prices across suppliers.

4.6.3 A Networked Market with Capacity Constraints

One advantage EC has over all the other pricing schemes is its generality. Specifically, EC can be applied to networked markets. In this section, we divide a single market with a fixed total demand 60 as described earlier into one market with only med tech suppliers and the other one with the smokestack and high-tech suppliers. The cost functions of the suppliers are the same as defined earlier, i.e., linear plus startup cost. As pictured in Figure 4.6, these two markets are connected via a flow capacity constraint. We consider two different cases of non-uniform marginal pricing and uniform marginal pricing for these two markets. Figure 4.7 shows how total payments, total uplifts and flow between these two connected markets vary as flow capacity increases for nonuniform and uniform marginal pricing settings. The results show that the total payments and total uplifts decrease as more flow is allowed between these two markets until it reaches the demand of one market, which means one market alone meets the total demand. Allowing non-uniform pricing does not further reduce total payments, as total payments are minimal and equal the total costs. However, it helps reduce total uplifts, as we can see in Figure 4.7b.



(a) Total payments as a function of flow capacity.

(b) Total uplifts as a function of flow capacity.

(c) Flow between two markets as a function of flow capacity.

Figure 4.7: An example of two connected markets with a constraint on the flow capacity.

4.7 Concluding Remarks

We study the problem of pricing in single and networked markets with non-convex costs. Our key contribution is the proposal of a novel scheme, Equilibrium-Constrained (EC) pricing, which optimizes for the allocations and price parameters at the same time, while imposing the equilibrium conditions as constraints. Our pricing framework is general in the sense that: (i) it can be used for pricing general non-convex cost functions, (ii) it allows for using general price classes, (iii) can be computed in polynomial-time regardless of the source of the non-convexities, and (iv) it extends easily to networked markets.

This work opens up a variety of important directions for future work. First, as this framework enables one to use general price classes, it would be interesting to apply it to specific classes of price functions (e.g., quadratic plus uplift, piece-wise, etc.) and characterize the solution theoretically and/or numerically. One can then investigate the potential trade-offs between the complexity of the class and the economic efficiency or the uniformity of the price. Second, since electricity markets are an important application of the pricing problem studied here, it would be interesting to evaluate the proposed scheme in practical settings for electricity markets. Our preliminary exploration shows that we can achieve more efficient (lower total payments) and less discriminatory (lower uplifts) prices with, for instance, piece-wise linear functions. More evaluations in large-scale, practical settings should be carried out in order to evaluate the potential of deployment. Another important direction to pursue is the extension of our results to networked markets with more general network structures. Our algorithm currently applies to networks with bounded tree-width; however beyond such networks, new ideas are needed. Finally, our proposed pricing scheme has broader implications for non-convex optimization problems as well. In the convex setting, dual prices are crucial for the development

of distributed optimization algorithms, but such approaches have not been possible in non-convex settings due to the lack of pricing rules with the desirable properties laid out in Section 4.2.2. It is now possible to explore whether EC prices can be used as the basis for distributed optimization algorithms in the non-convex setting.

4.A Supplement to Section 4.3.1

In this section, we formally prove the reduction of the optimization problem for the class of linear-plus-uplift functions to (4.6), and then show Propositions 18 and 19.

4.A.1 Reduction

Here, we show that for the class of linear-plus-uplift price functions $p(q_i; \lambda, u_i, \hat{q}_i) = \lambda q_i + u_i \mathbb{1}_{q_i=\hat{q}_i}$, one can assume $\hat{q}_i^* = q_i^*$ without loss of generality, and therefore the optimization problem (4.5) reduces to (4.6) for this class. The optimization problem (4.5) for price function $p(q_i; \lambda, u_i, \hat{q}_i) = \lambda q_i + u_i \mathbb{1}_{q_i=\hat{q}_i}$, $\lambda, u_1, \ldots, u_n \geq 0$, is as follows

$$p^*_{\text{uplift}} = \min_{\substack{q_1,\ldots,q_n \\ \lambda \geq 0 \\ u_1,\ldots,u_n \geq 0 \\ \hat{q}_1,\ldots,\hat{q}_n}} \quad \sum_{i=1}^{n} (\lambda q_i + u_i \mathbb{1}_{q_i=\hat{q}_i}) \tag{4.25a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d \tag{4.25b}$$

$$\lambda q_i + u_i \mathbb{1}_{q_i=\hat{q}_i} - c_i(q_i) \geq 0, \quad i = 1, \ldots, n \tag{4.25c}$$

$$\lambda q_i + u_i \mathbb{1}_{q_i=\hat{q}_i} - c_i(q_i) \geq \max_{q_i' \neq q_i} \lambda q_i' + u_i \mathbb{1}_{q_i'=\hat{q}_i} - c_i(q_i'), \quad i = 1, \ldots, n \tag{4.25d}$$

The following lemma shows that this optimization problem can be reduced to (4.6), and the optimal uplifts of (4.6) are no larger than those of (4.25).

Lemma 22. *Given any solution* $(\boldsymbol{q}^*, \lambda^*, \boldsymbol{u}^*, \hat{\boldsymbol{q}}^*)$ *to the optimization problem* (4.25)*,* $(\boldsymbol{q}^*, \lambda^*, \underline{\boldsymbol{u}}, \boldsymbol{q}^*)$ *is also a solution, where*

$$\underline{u}_i = \begin{cases} u_i^*, & \text{if } \hat{q}_i^* = q_i^* \\ 0, & \text{o.w.} \end{cases}.$$

Proof of Lemma 22. Let us first show the feasibility of $(\boldsymbol{q}^*, \lambda^*, \underline{\boldsymbol{u}}, \boldsymbol{q}^*)$. For any i such that $\hat{q}_i^* \neq q_i^*$, we have that

$$\begin{aligned} &\lambda^* q_i^* - c_i(q_i^*) \geq 0 \\ &\lambda^* q_i^* - c_i(q_i^*) \geq \max_{q_i' \neq q_i^*} \lambda^* q_i' + u_i^* \mathbb{1}_{q_i'=\hat{q}_i^*} - c_i(q_i') \geq \max_{q_i' \neq q_i^*} \lambda^* q_i' - c_i(q_i'), \end{aligned}$$

which implies

$$\begin{aligned} &\lambda^* q_i^* + \underline{u}_i^* \mathbb{1}_{q_i^*=q_i^*} - c_i(q_i^*) \geq 0 \\ &\lambda^* q_i^* + \underline{u}_i^* \mathbb{1}_{q_i^*=q_i^*} - c_i(q_i^*) \geq \max_{q_i' \neq q_i^*} \lambda^* q_i' + \underline{u}_i^* \mathbb{1}_{q_i'=\hat{q}_i^*} - c_i(q_i'), \end{aligned}$$

because $\underline{u}_i^* = 0$. Therefore $(\boldsymbol{q}^*, \lambda^*, \underline{\boldsymbol{u}}, \boldsymbol{q}^*)$ is feasible.

The objective value of $(\boldsymbol{q}^*, \lambda^*, \underline{\boldsymbol{u}}, \boldsymbol{q}^*)$ is

$$\begin{aligned}\sum_{i=1}^{n}(\lambda^* q_i^* + \underline{u}_i) &= \sum_{i:\hat{q}_i^*=q_i^*}(\lambda^* q_i^* + u_i^*) + \sum_{i:\hat{q}_i^*\neq q_i^*}\lambda^* q_i^* \\ &= \sum_{i=1}^{n}(\lambda^* q_i^* + u_i^* \mathbb{1}_{q_i^*=\hat{q}_i^*}),\end{aligned}$$

which is the same as that of $(\boldsymbol{q}^*, \lambda^*, \boldsymbol{u}^*, \hat{\boldsymbol{q}}^*)$, and is therefore optimal. □

Based on this lemma, the optimization problem (4.25) can be reduced to (4.6).

4.A.2 Closed-Form Solutions

Proof of Proposition 18. In the optimization problem (4.6), the order of variables in the minimizations does not matter, and further, for every fixed $q_1, \ldots, q_n$ and λ, the minimization over each u_i can be done separately. Therefore, this program can be massaged into the following form

$$p_{\text{uplift}}^* = \min_{q_1,\ldots,q_n} \quad \left(\min_{\lambda\geq 0} \sum_{i=1}^{n} g_i(q_i;\lambda)\right) \tag{4.26a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d, \tag{4.26b}$$

where

$$g_i(q_i;\lambda) = \min_{u_i\geq 0} \quad \lambda q_i + u_i \tag{4.27a}$$

$$\text{s.t.} \quad \lambda q_i + u_i - c_i(q_i) \geq 0, \tag{4.27b}$$

$$\lambda q_i + u_i - c_i(q_i) \geq \max_{q_i'\neq q_i} \lambda q_i' - c_i(q_i'). \tag{4.27c}$$

for all $i = 1, \ldots, n$. Constraints (4.27b) and (4.27c) can be expressed as

$$\begin{aligned}&\lambda q_i + u_i \geq c_i(q_i),\\ &\lambda q_i + u_i \geq c_i(q_i) + \max_{q_i'\neq q_i} \lambda q_i' - c_i(q_i').\end{aligned}$$

It follows that

$$g_i(q_i;\lambda) = \lambda q_i + u_i^* = c_i(q_i) + \max\left\{0, \max_{q_i'\neq q_i} \lambda q_i' - c_i(q_i')\right\}.$$

which is, of course, a function of λ and q_i. Therefore, we have

$$\min_{\lambda \geq 0} \sum_{i=1}^{n} g_i(q_i; \lambda) \;=\; \sum_{i=1}^{n} c_i(q_i),$$

and the minimizers λ^* are all values λ for which $\max_{q_i' \neq q_i} \lambda q_i' - c_i(q_i') \leq 0$, which are exactly the elements of $\Lambda = \{\lambda \geq 0 \mid \lambda q \leq c_i(q),\ \forall q, \forall i\}$ (Figure 4.1 provides a pictorial description of these values.) Finally, we have the last minimization, which is

$$\min_{q_1,\ldots,q_n} \quad \sum_{i=1}^{n} c_i(q_i) \tag{4.28a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d \tag{4.28b}$$

and therefore has $q_i^* = q_i^0\ \forall i$ as its optimizer. We also have $u_i^* = c_i(q_i^*) - \lambda^* q_i^*,\ \forall i$. □

Proof of Proposition 19. The steps of the proof are exactly the same as in the previous one, except that the additional minimizer picks the λ with the smallest total uplift $\sum_{i=1}^{n} u_i(\lambda)$, which corresponds to the largest element of Λ. □

4.B Supplement to Section 4.3.2

In this section, we prove Theorem 20, in two parts. First, we show that there exist finite sets $Q, \mathcal{A}', \mathcal{B}'$ for which Algorithm 1 finds an ϵ-approximate solution, and we quantify the sizes of these sets as a function of ϵ. In the second part, we analyze the running time of Algorithm 1.

4.B.1 ϵ-Accuracy

Let us first state a simple but useful lemma.

Lemma 23 (δ-discretization)**.** *Given a set $C \subseteq [\underline{L_1}, \overline{L_1}] \times \cdots \times [\underline{L_k}, \overline{L_k}]$, for any $\delta > 0$, there exists a finite set C' such that*

$$\forall z \in C,\ \exists z' \in C' \text{ s.t. } \|z - z'\|_\infty \leq \delta,$$

and further, C' contains at most V/δ^k points, where $V = \prod_{i=1}^{k}(\overline{L_i} - \underline{L_i})$ is a constant (the volume of the box). C' is said to be a δ-discretization of C.

Let Q, $\mathcal{A}'$, and $\mathcal{B}'$ denote some δ-discretizations of sets $[0, d]$, $\mathcal{A}$ and $\mathcal{B}$, respectively. In other words, for every $q \in [0, d]$, $\alpha \in \mathcal{A}$, and $\beta \in \mathcal{B}$, there exist $q' \in Q$, $\alpha' \in \mathcal{A}'$, and $\beta' \in \mathcal{B}'$, such that $|q - q'| \leq \delta$, $\|\alpha - \alpha'\|_\infty \leq \delta$, and $\|\beta - \beta'\|_\infty \leq \delta$. We can combine all these inequalities as

$$\|(q, \alpha, \beta) - (q', \alpha', \beta')\|_\infty \leq \delta.$$

On the other hand, given that the cost function $c_i(.)$ for each i is Lipschitz on each continuous piece of its domain, there exists a positive constant K_i such that $|c_i(q) - c_i(q')| \leq K_i|q - q'|$, which implies

$$|c_i(q) - c_i(q')| \leq K_i\delta. \tag{4.29}$$

Similarly, Lipschitz continuity of $p(.;.)$ implies existence of a positive constant K such that $|p(q, \alpha, \beta) - p(q', \alpha', \beta')| \leq K\|(q, \alpha, \beta) - (q', \alpha', \beta')\|_\infty$, which yields

$$|p(q, \alpha, \beta) - p(q', \alpha', \beta')| \leq K\delta. \tag{4.30}$$

Using Eqs. (4.29),(4.30), we can see that for any solution $q_1^*, \ldots, q_n^*, \alpha^*, \beta_1^*, \ldots, \beta_n^*$ to optimization (4.5), there exists a point $q_1, \ldots, q_n, \alpha, \beta_1, \ldots, \beta_n$ with $q_1, \ldots, q_n \in Q$, $\alpha \in \mathcal{A}'$ and $\beta \in \mathcal{B}'$, for which constraints (4.5c) and (4.5d) are violated at most by

$(K + K_i)\delta$ and $(2K + 2K_i)\delta$, respectively, and the objective is larger than p^* at most by $nK\delta$. As a result, this point will be an ϵ-approximate solution if

$$(K + K_i)\delta \leq \epsilon \quad \forall i, \tag{4.31}$$

$$2(K + K_i)\delta \leq \epsilon \quad \forall i, \tag{4.32}$$

$$nK\delta \leq n\epsilon. \tag{4.33}$$

These constraints altogether enforce an upper bound on the value of δ as

$$\delta \leq C\epsilon,$$

for some constant C. Therefore if we pick

$$\delta = \frac{d}{\lceil * \rceil \frac{d}{C\epsilon}}, \tag{4.34}$$

our algorithm is guaranteed to encounter an ϵ-approximate solution while enumerating the points, and $Q = \{0, \delta, 2\delta, \ldots, d\}$ is a valid δ-discretization for $[0, d]$, which has $N_q = \lceil * \rceil \frac{d}{C\epsilon} + 1 = O\left(\frac{1}{\epsilon}\right)$ points. The nice thing about this particular choice of δ is that now d can be written as a sum of n elements in Q (because all the elements, including d, are multiples of δ), which allows us to satisfy the Market Clearing condition exactly. Based on Lemma (23), $\mathcal{A}'$ and $\mathcal{B}'$ contain $N_\alpha = O\left(\frac{1}{\delta^{l_1}}\right) = O\left(\frac{1}{\epsilon^{l_1}}\right)$ and $N_\beta = O\left(\frac{1}{\delta^{l_2}}\right) = O\left(\frac{1}{\epsilon^{l_2}}\right)$ points.

Finally, if there are any discontinuities in the cost or price functions, we can simply add them to our discrete sets Q, $\mathcal{A}'$, and $\mathcal{B}'$, and since there are at most a finite number of them, the sizes of the sets remain in the same order, i.e., $N_q = O\left(\frac{1}{\epsilon}\right)$, $N_\alpha = O\left(\frac{1}{\epsilon^{l_1}}\right)$, and $N_\beta = O\left(\frac{1}{\epsilon^{l_2}}\right)$. Next, we calculate the time complexity of Algorithm 1 running on these discrete sets.

4.B.2 Run-Time Analysis

In this section, we show that Algorithm 1 has a time complexity of $O\left(n(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$. For every fixed α, we have the following computations:

1. The leaves: We need to compute $g_i(q; \alpha)$ for every i and every $q \in Q$. Computing each $g_i(q; \alpha)$ (i.e., for fixed i, q, α) takes $O(N_\beta N_q)$. The reason for that is we have to search over all $\beta_i \in B'$, and for each one there are

$N_q + 1$ constraints to check. More explicitly, we need to (a) check $O(N_\beta N_q)$ constraints, (b) compute N_β objectives, and (c) find the minimum among those N_β values. All these steps together take $O(N_\beta N_q)$, and repeating for every i and q makes it $O(nN_\beta N_q^2)$.

2. The intermediate nodes: In each new level, there are at most half as many (+1) nodes as in the previous level. For each node i in this level, we need to compute $g_i(q;\alpha)$ for every $q \in Q$. For every fixed q, there are $O(N_q)$ possible pairs of (q_j, q_k) that add up to q, and therefore we need to (a) sum $O(N_q)$ pairs of objective values, and (b) find the minimum among them, which take $O(N_q)$. Hence, the computation for each node takes $O(N_q^2)$. There are $O(\frac{n}{2} + \frac{n}{4} + \cdots + 2) = O(n)$ intermediate nodes in total, and therefore the total complexity of this part is $O(nN_q^2)$.

3. The root: Finally at the root, we need to compute $g_{\text{root}}(d;\alpha)$. There are N_q possible pairs of (q_j, q_k) that add up to d. Therefore, we need to compute N_q sums, and find the minimum among the resulting N_q values, which takes $O(N_q)$.

Putting the pieces together, the computation for all values of α takes $N_\alpha \times \Big(O(nN_\beta N_q^2) + O(nN_q^2) + O(N_q)\Big)$, which in turn is $O(nN_\alpha N_\beta N_q^2)$. Finally, finding the minimum among the N_α values simply takes $O(N_\alpha)$.

The backward procedure, which finds the quantities q_i and the parameters β_i, takes just $O(n)$, since it is just a substitution for every node. As a result, the total running time is $O(nN_\alpha N_\beta N_q^2)$, which based on the first part (Section 4.B.1) is $O\left(n(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$.

4.B.3 Remark on the ϵ-Approximation

As mentioned at the end of Section 4.3.2, if one requires the total payment in Definition 1 to be at most ϵ (rather than $n\epsilon$) away from the optimal p^*, the running time of our algorithm will still be polynomial in both n and $1/\epsilon$, i.e., $O\left(n^3(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$. To see that, notice in this case (4.31) and (4.32) remain the same, and (4.33) changes to $nK\delta \le \epsilon$. Therefore, the upper bound enforced by the constraints will be $\delta \le \frac{C\epsilon}{n}$, for some constant C. In this case, our choice of δ would be $\delta = \frac{d}{\lceil * \rceil \frac{dn}{C\epsilon}}$, and hence $N_q = O\left(\frac{n}{\epsilon}\right)$. N_α and N_β remain the same as before. The running time is $O(nN_\alpha N_\beta N_q^2)$, as computed previously, which in this case would be $O\left(n^3(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$.

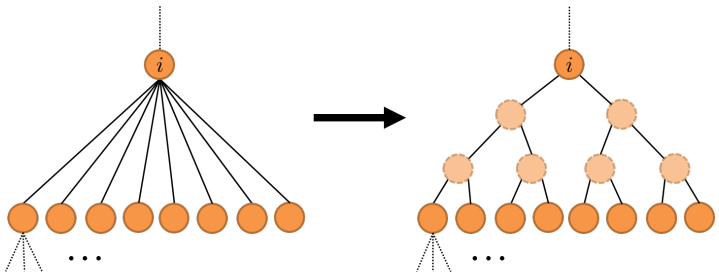


Figure 4.8: The transformation of an arbitrary-degree tree to a binary tree.

4.C Supplement to Section 4.4

In this section, we first show the transformation of the problem on a tree to one on a binary tree, and then prove Theorem 21.

4.C.1 Transformation into Binary Tree

Lemma 24. *Given any tree with n nodes (suppliers), there exists a binary tree with additional nodes which has the same solution* $(q_i^*, \dots, q_n^*, \alpha^*, \beta_1, \dots, \beta_n)$ *for those nodes as the original network. The binary tree has* $O(n)$ *nodes.*

Proof. Take any node i that has $k_i > 2$ children. For any two children, introduce a dummy parent node. For any two dummy parent nodes, introduce a new level of dummy parent nodes. Continue this process until there are 2 or less nodes in the uppermost layer, and then connect them to node i (see Fig. 4.8). The capacities of the lines immediately connected to the children are the same as those in the original graph. The capacities of the new lines are infinite.

The total number of introduced dummy nodes by this procedure is

$$O(\frac{k_i}{2} + \frac{k_i}{4} + \cdots + 2) = O(k_i).$$

Since there are $1 + k_1 + k_2 + \cdots + k_n = n$ nodes in total in the original tree, the number of introduced additional nodes is $O(k_1 + \cdots + k_n) = O(n)$. Therefore the total number of nodes in the new (binary) tree is $O(n)$. □

4.C.2 Proof of Theorem 21

Most of the proof is similar to the one presented in Section 4.B. For this reason, we only highlight the main points. The proof consists of ϵ-accuracy and run-time, as before.

4.C.2.1 ϵ-Accuracy

Let $Q_1, \ldots, Q_n, F_1, \ldots, F_n, \mathcal{A}', \mathcal{B}'$ denote some δ-discretizations of sets $[0, d_1 + \overline{f_{\text{ch}_1(1)}} + \overline{f_{\text{ch}_2(1)}} - \underline{f_1}], \ldots, [0, d_n + \overline{f_{\text{ch}_1(\text{n})}} + \overline{f_{\text{ch}_2(\text{n})}} - \underline{f_n}]$, $[\underline{f_1}, \overline{f_1}], \ldots, [\underline{f_n}, \overline{f_n}]$, $\mathcal{A}, \mathcal{B}$, respectively. Note that if any line capacities are infinite, the intervals can be replaced by $[0, \sum_{i=1}^{n} d_i]$ instead. Similar as in Section 4.B, the constraints enforce an upper bound on the value of δ as $\delta \leq C\epsilon$, for some constant C. Based on Lemma (23), the sizes of the sets will be $N_{q_i} = O\left(\frac{1}{\epsilon}\right)\ \forall i$, $N_{f_i} = O\left(\frac{1}{\epsilon}\right)\ \forall i$, $N_\alpha = O\left(\frac{1}{\epsilon^{l_1}}\right)$, and $N_\beta = O\left(\frac{1}{\epsilon^{l_2}}\right)$

4.C.2.2 Run-Time Analysis

For every fixed α, the run-time of the required computations is as follows.

1. The time complexity of computing $g_i(q_i; \alpha)$ for each node i and each fixed value of q_i is $O(N_\beta N_{q_i})$. Therefore, computing it for all nodes and all values takes $O(nN_\beta N_q^2)$.

2. Computing $h_i(f_i; \alpha)$ for each node i and each fixed value of f_i takes $O(N_f^2)$, because there are $O(N_f) \times O(N_f)$ pairs of values for $(f_{\text{ch}_1(i)}, f_{\text{ch}_2(i)})$ (q_i is automatically determined as the closest point in Q_i to $d_i + f_{\text{ch}_1(i)} + f_{\text{ch}_2(i)} - f_i$). Therefore, its overall computation for all nodes and all values takes $O(nN_f^3)$.

As a result, the overall computation takes $N_\alpha \times \left(O\left(nN_\beta N_q^2\right) + O\left(nN_f^3\right)\right)$, which is $O\left(n(\frac{1}{\epsilon})^{l_1+l_2+2}\right) + O\left(n(\frac{1}{\epsilon})^{l_1+3}\right)$, or equivalently $O\left(n(\frac{1}{\epsilon})^{l_1+\max\{l_2,1\}+2}\right)$.

Chapter 5

MANAGING AGGREGATORS IN THE SMART GRID

[1] Navid Azizan et al. "Opportunities for Price Manipulation by Aggregators in Electricity Markets". In: *SIGMETRICS Performance Evaluation Review* 44.2 (2016), pp. 49–51. ISSN: 0163-5999. DOI: 10.1145/3003977.3003995.

[2] Navid Azizan et al. "Opportunities for Price Manipulation by Aggregators in Electricity Markets". In: *IEEE Transactions on Smart Grid* 9.6 (2018), pp. 5687–5698. DOI: 10.1109/TSG.2017.2694043.

Aggregators of distributed generation are playing an increasingly crucial role in the integration of renewable energy in power systems. However, the intermittent nature of renewable generation makes market interactions of aggregators difficult to monitor and regulate, raising concerns about potential market manipulation by aggregators. In this chapter, we study this issue by quantifying the profit an aggregator can obtain through strategic curtailment of generation in an electricity market. We show that, while the problem of maximizing the benefit from curtailment is hard in general, efficient algorithms exist when the topology of the network is radial (acyclic). Further, we highlight that significant increases in profit are possible via strategic curtailment in practical settings.

5.1 Introduction

Increasing the penetration of distributed, renewable energy resources into the electricity grid is a crucial part of building a sustainable energy landscape. To date, the entities that have been most successful at promoting and facilitating the adoption of renewable resources have been *aggregators*, e.g., SolarCity, Tesla, Enphase, Sunnova, SunPower, and ChargePoint [47, 110, 209]. These aggregators install and manage rooftop solar installations as well as household energy storage devices and electric vehicle charging systems. Some have fleets with upwards of 800 MW distributed energy resources [192, 1], and the market is expected to triple in size by 2020 [133, 104].

Aggregators play a variety of important roles in the construction of a sustainable grid. First, and foremost, they are on the front lines of the battle to promote installation of rooftop solar and household energy storage, pushing for widespread adoption

of distributed energy resources by households and businesses. Second, and just as importantly, they provide a single interface point where utilities and Independent System Operators (ISOs) can interact with a fleet of distributed energy resources across the network in order to obtain a variety of services, from renewable generation capacity to demand response. This service is crucial for enabling system operators to manage the challenges that result from unpredictable, intermittent renewable generation, e.g., wind and solar.

However, in addition to the benefits they provide, aggregators also create new challenges – both from the perspective of the aggregator and the perspective of the system operator. On the side of the aggregator, the management of a geographically diverse fleet of distributed energy resources is a difficult algorithmic challenge. On the side of the operator, the participation of aggregators in electricity markets presents unique challenges in terms of monitoring and mitigating the potential of exercising market power. In particular, unlike traditional generation resources, the ISO cannot verify the availability of the generation resources of aggregators. While the repair schedule of a conventional generator can be made public, the downtime of a solar generation plant and the times when solar generation is not available cannot be scheduled or verified after the fact. Thus, aggregators have the ability to strategically curtail generation resources without the knowledge of the ISO, and this potentially creates significant opportunities for them to manipulate prices.

These issues are particularly salient given current proposals for distribution systems. Distribution systems (which are typically radial networks) are heavily impacted by the introduction of distributed energy resources. As a result, there are a variety of current proposals to start distribution-level power markets (see, for example [105] [106]), operated by Distribution System Operators (DSOs). A future grid may even involve a hierarchy of system operators dealing with progressively larger areas, net load and net generation. In such a scenario, aggregators could end up having a significant proportion of the market share, and such markets may be particularly vulnerable to strategic bidding practices of the aggregators. Thus, understanding the potential for these aggregators to exercise market power is of great importance, so that regulatory authorities can take appropriate steps to mitigate it as needed.

5.1.1 Summary of Contributions

This chapter addresses both the algorithmic challenge of managing an aggregator and the economic challenge of measuring the potential for an aggregator to manipulate

prices. Specifically, this work provides a new algorithmic framework for managing the participation of an aggregator in electricity markets, and uses this framework to evaluate the potential for aggregators to exercise market power. To those ends, the chapter makes three main contributions.

First, we introduce a new model for studying the market behavior of aggregators of distributed generation (renewables) in the real-time market.

Second, we quantify opportunities for price manipulation (via strategic curtailment) by the aggregators. Our results highlight that, in practical scenarios, strategic curtailment can have a significant impact on prices, and yield much higher profits for the aggregators. In particular, the prices can be impacted up to a few tens of $/MWh in some cases, and there is often more than 25% higher profit, even with curtailments limited to 1%.

Third, we provide a novel algorithm for managing the participation of an aggregator in the market. The problem is NP-hard in general and is a bilevel quadratic program, which is notoriously difficult in practice. However, we develop an efficient algorithm that can be used by the aggregators in radial networks to approximate the optimal curtailment strategy and maximize their profit (Section 5.5). Note that the algorithm is not just relevant for aggregators; it can also be used by the operator to assess the potential for strategic curtailment. The key insight in the algorithm is that the optimization problem can be decomposed into "local" pieces and be solved approximately using a dynamic programming over the graph. We also provide an exact algorithm for the case of single-bus aggregators in general networks.

Further, our results expose a connection between the profit achievable via curtailment and a new market power measure introduced in [221], which is discussed in Appendix 5.A.

5.1.2 Related Work

This chapter connects to, and builds on, work in four related areas: 1) quantifying and mitigating market power, 2) cyber-attacks in the grid, 3) algorithms for managing distributed energy resources, and 4) algorithms for bilevel programs.

5.1.2.1 Quantifying Market Power in Electricity Markets

There is a large volume of literature that focuses on identifying and measuring market power for generators in an electricity market, see [203] for a recent survey.

Early works on market power analysis, emerged from microeconomic theory, suggest measures that ignore transmission constraints. For example, [48] introduced the *pivotal supplier index* (PSI), which is a binary value indicating whether the capacity of a generator is larger than the supply surplus, and [184] later refined PSI by proposing *residential supply index* (RSI). RSI is used by the California ISO to assure price competitiveness [49]. The electricity reliability council of Texas uses the *element competitiveness index* (ECI) [71], which is based on the *Herfindahl-Hirschmann index* (HHI) [182].

Market power measures considering transmission constraints have emerged more recently. Some examples include, e.g., [180, 41, 161, 52, 218], and [219]. Interested readers can refer to [42], which proposes a functional measure that unifies the structural indices measuring market power on a transmission constrained network in the previous work.

In contrast to the large literature discussed above, the literature focused on market power of renewable generation producers is limited. Existing works such as [221] and [202] study market power of wind power producers ignoring transmission constraints. The key differentiator of the work in this chapter is that the use of the Locational Marginal Price (LMP) framework, which is standard practice in the electricity market [162, 226], allows this work to offer insight about market power of aggregators when transmission capacity is limited.

5.1.2.2 Cyber-Attacks in the Grid

The model and analysis in this chapter is also strongly connected to the cyber security research community, which has studied how and when a malicious party can manipulate the spot price in electricity markets by compromising the state measurement of the power grid via false data injection [35, 36, 216, 217, 136].

In particular, [216, 217] shows that if a malicious party can corrupt sensor data, then it can create an arbitrage opportunity. Further, [35] shows that such attacks can impact both the real time spot price and future prices by causing line congestions.

In this work, we do not allow aggregators to corrupt the state measurements of the power system, rather we consider a perfectly legal approach for price manipulation: strategic curtailment. However, strategic curtailment in the *ex-post* market can gain extra profit to the detriment of the power system, which is a similar mechanism to those highlighted in cyber attack literature. Technically, the work in this chapter makes

significant algorithmic contributions to the cyber-attack literature. In particular, the papers mentioned above focus on algorithmic heuristics and do not provide formal guarantees. In contrast, our work presents a polynomial-time algorithm that provably maximizes the profit of the aggregator.

5.1.2.3 Algorithms for Managing Distributed Energy Resources

There has been much work studying optimal strategies for managing demand response and distributed generation resources to offer regulation services to the power grid. This work covers a variety of contexts. For example, researchers have studied frequency regulation [132][92] and voltage regulation (or volt-VAR control) [223][12]. A separate line of work has been work on designing incentives to encourage distributed resources to provide services to the power grid [148][59]. However, the current chapter is distinct from all the work above in that we study strategic behavior by an aggregator of distributed resources. Prior work does not model the strategic manipulation of prices by the aggregator.

5.1.2.4 Algorithms for Bilevel Programs

The optimization problem that the strategic aggregator solves is a bilevel program, since the objective (aggregator's profit) depends on the locational prices (LMPs). The LMPs are constrained to be equal to optimal dual variables arising from economic dispatch-based market clearing procedure. These types of problems have been extensively studied in the literature, and fall under the class of Mathematical Programs with Equilibrium Constraints (MPECs) [73]. Even if the optimization problems at the two levels are linear, the problem is known to be NP-hard [34]. Global optimization algorithms [84] can be used to solve these problems to arbitrary accuracy (compute a lower bound on the objective within a specified tolerance of the global optimum). However, these algorithms use a spatial branch and bound strategy, and can take exponential time in general. In contrast, solvers like PATH [63], while practically efficient for many problems, are only guaranteed to find a local optimum. In this chapter, we show that for tree-structured networks (distribution networks), an ϵ-approximation of the global optimum can be computed in time linear in the size of the network and polynomial in $\frac{1}{\epsilon}$.

5.2 System Model

In this section, we define the power system model that serves as the basis for the chapter and describe how we model the way the Independent System Operator (ISO)

computes the Locational Marginal Prices (LMPs). Locational marginal pricing is adopted by the majority of power markets in the Unites States [226], and our model is meant to mimic the operation of two-stage markets like ISO New England, PJM Interconnection, and Midcontinent ISO, that use ex-post pricing strategy for correcting the ex-ante prices [162, 226].

5.2.1 Preliminaries

We consider a power system with n nodes (buses) and t transmission lines. The generation and load at node i are denoted by p_i and d_i respectively, with $\mathbf{p} = \left[p_1, \ldots, p_n\right]^T$ and $\mathbf{d} = \left[d_1, \ldots, d_n\right]^T$. We use $[n]$ to denote the set of buses $\{1, \ldots, n\}$.

The focus of this chapter is on the behavior of an aggregator in the real-time market, which owns generation capacity, possibly at multiple nodes. We assume that the aggregator has the ability to curtail generation, e.g., by curtailing the amount of wind/solar generation or by not calling on demand response opportunities, without penalty. This is because in many of today's markets, the renewable generation (e.g., solar) can be sold at the real-time price without having to commit to the ex-ante market (see for example CAISO Participating Intermittent Resource Program (PIRP) [140]). Let $N_a \subseteq [n]$ be the nodes where the aggregator has generation and denote its share of generation at node $i \in N_a$ by p_i^a (out of p_i). The curtailment of generation at this node is denoted by α_i, where $0 \leq \alpha_i \leq p_i^a$. We define our model for the decision-making process of the aggregator with respect to curtailment in Section 5.3.

Together, the net generation delivered to the grid is represented by $\mathbf{p} - \boldsymbol{\alpha}$, where $\alpha_j = 0 \ \forall j \notin N_a$. The flow of lines is denoted by $\mathbf{f} = \left[f_1, \ldots, f_t\right]^T$, where f_l represents the flow of line l: $\mathbf{f} = \mathbf{G}(\mathbf{p} - \boldsymbol{\alpha} - \mathbf{d})$, where $\mathbf{G} \in \mathbb{R}^{t \times n}$ is the matrix of generation shift factors [186]. We also define $\mathbf{B} \in \mathbb{R}^{n \times t}$ as the link-to-node incidence matrix that transforms line flows back to the net injections as $\mathbf{p} - \boldsymbol{\alpha} - \mathbf{d} = \mathbf{B}\mathbf{f}$.

5.2.2 Real-Time Market Price

For every dispatch interval, the ISO obtains the current values of generation, demands, and flows from the state estimator, in real time. Based on this information, it solves a constrained optimization problem for market clearing. The objective of the optimization is to minimize the total cost of the network, based on the current state of the system. The ex-post LMPs are announced as a function of the optimal Lagrange multipliers of this optimization. Mathematically, the following program has to be

solved.

$$\begin{aligned}
&\underset{\mathbf{f}}{\text{minimize}} && \mathbf{c^T B f} && \text{(5.1a)}\\
&\text{subject to} \\
&\boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+ : && \underline{\boldsymbol{\Delta}\mathbf{p}} \le \mathbf{Bf} - \mathbf{p} + \boldsymbol{\alpha} + \mathbf{d} \le \overline{\boldsymbol{\Delta}\mathbf{p}} && \text{(5.1b)}\\
&\boldsymbol{\mu}^-, \boldsymbol{\mu}^+ : && \underline{\mathbf{f}} \le \mathbf{f} \le \bar{\mathbf{f}} && \text{(5.1c)}\\
&\boldsymbol{\nu} : && \mathbf{f} \in \text{range}(\mathbf{G}) && \text{(5.1d)}
\end{aligned}$$

In the above, c_i is the offer price for the generator i. f_l is the desired flow of line l, and $\mathbf{Bf} = \mathbf{p} + \boldsymbol{\Delta}\mathbf{p} - \boldsymbol{\alpha} - \mathbf{d}$, where Δp_i is the desired amount of change in the generation of node i. Constraint (5.1b) enforces the upper and lower limits on the change of generations, and constraint (5.1c) keeps the flows within the line limits. In practice, $\underline{\Delta p_i}$ and $\overline{\Delta p_i}$ are usually set to be a constant value for all i (e.g., $\underline{\Delta p_i} = -2$ and $\overline{\Delta p_i} = 0.1$, $\forall i$ [166, p. 100]). The last constraint ensures that f_l are valid flows, i.e., $\mathbf{f} = \mathbf{G}\tilde{\mathbf{p}}$ for some generation $\tilde{\mathbf{p}}$. Variables $\boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+ \in \mathbb{R}^n_+$, $\boldsymbol{\mu}^-, \boldsymbol{\mu}^+ \in \mathbb{R}^t_+$, and $\boldsymbol{\nu} \in \mathbb{R}^{t-\text{rank}(G)}$ denote the Lagrange multipliers (dual variables) corresponding to constraints (5.1b), (5.1c), and (5.1d).

Note that the ISO does not physically redispatch the generations, and the optimal values of the above program are just the desired values. In fact, by announcing the (ex-post) LMPs, the ISO provides incentives for the generators to adjust their generation according to its goals [226].

Definition 3. *The **ex-post locational marginal price (LMP)** of node i at curtailment level of $\boldsymbol{\alpha}$, denoted by $\lambda_i(\boldsymbol{\alpha})$, is*

$$\lambda_i(\boldsymbol{\alpha}) = c_i + \lambda_i^+(\alpha) - \lambda_i^-(\alpha). \tag{5.2}$$

We assume that the LMPs are unique. Non-uniqueness of LMPs happens only under very special degenerate conditions, and can be fixed in practice by adding a quadratic penalty term to the objective to make it convex [50].

5.3 The Market Behavior of the Aggregator

The key feature of our model is the behavior of the aggregator. As mentioned before, aggregators have generation resources at multiple locations in the network and can often curtail generation resources without the knowledge of the ISO. Of course, such curtailment may not be in the best interest of the aggregator, since it means offering less generation to the market. But, if through curtailment, prices can be impacted,

then the aggregator may be able to receive higher prices for the generation offered or make money through arbitrage of the price differential.

To quantify the profit that the aggregator makes due to the curtailment, let us take a look at the total revenue in different production levels.

Definition 4. *We define the **curtailment profit (CP)** as the change in profit of the aggregator as a result of curtailment:*

$$\gamma(\boldsymbol{\alpha}) = \sum_{i \in N_a} \left(\lambda_i(\boldsymbol{\alpha}) \cdot (p_i^a - \alpha_i) - \lambda_i(0) \cdot p_i^a \right). \tag{5.3}$$

Note that the curtailment profit can be positive or negative in general. We say a curtailment level $\boldsymbol{\alpha} > 0$ is profitable if $\gamma(\boldsymbol{\alpha})$ is strictly positive.

The curtailment profit is important for understanding when it is beneficial for the aggregators to curtail. Note that we are not concerned about the cost of generation here, as renewables have zero marginal cost. However, if there is a cost for generation, then that results in an additional profit during curtailment, which makes strategic curtailment more likely.

While our setup may seem divorced from the notion of market power, it turns out that there is a fundamental relationship between the curtailment profit introduced above and market power. See Appendix 5.A for details.

5.3.1 A Profit-Maximizing Aggregator

A natural model for a strategic aggregator is one that maximizes curtailment profit subject to LMPs and curtailment constraints. Since LMPs are the solution to an optimization problem themselves, the aggregator's problem is a bilevel optimization problem. In order to be able to express this optimization in an explicit form, let us first write the KKT conditions of the program (5.1).

Primal feasibility:

$$\underline{\boldsymbol{\Delta p}} \leq \mathbf{Bf} - \mathbf{p} + \boldsymbol{\alpha} + \mathbf{d} \leq \overline{\boldsymbol{\Delta p}} \tag{5.4a}$$

$$\underline{\mathbf{f}} \leq \mathbf{f} \leq \overline{\mathbf{f}} \tag{5.4b}$$

$$\mathbf{Hf} = \mathbf{0} \tag{5.4c}$$

Dual feasibility:

$$\boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+, \boldsymbol{\mu}^-, \boldsymbol{\mu}^+ \geq 0 \tag{5.4d}$$

Complementary slackness:

$$\lambda_i^+((Bf)_i - p_i + \alpha_i + d_i - \overline{\Delta p_i}) = 0, \quad i = 1, \ldots, n \tag{5.4e}$$

$$\lambda_i^-(\underline{\Delta p}_i - (Bf)_i + p_i - \alpha_i - d_i) = 0, \quad i = 1, \ldots, n \tag{5.4f}$$

$$\mu_l^+(f_l - \overline{f}_l) = 0, \quad l = 1, \ldots, t \tag{5.4g}$$

$$\mu_l^-(\underline{f}_l - f_l) = 0, \quad l = 1, \ldots, t \tag{5.4h}$$

Stationarity:

$$\mathbf{B}^T(\boldsymbol{c} + \boldsymbol{\lambda}^+ - \boldsymbol{\lambda}^-) + \boldsymbol{\mu}^+ - \boldsymbol{\mu}^- + \mathbf{H}^T\boldsymbol{\nu} = \mathbf{0}. \tag{5.4i}$$

Here $\mathbf{H} \in \mathbb{R}^{(t-\text{rank}(G))\times t}$, and the range of $\mathbf{G}$ is the nullspace of $\mathbf{H}$.

Using the KKT conditions derived above, the aggregator's problem can be formulated as follows.

$$\gamma^* = \underset{\boldsymbol{\alpha},\mathbf{f},\boldsymbol{\lambda}^-,\boldsymbol{\lambda}^+,\boldsymbol{\mu}^-,\boldsymbol{\mu}^+,\boldsymbol{\nu}}{\text{maximize}} \quad \gamma(\boldsymbol{\alpha}) \tag{5.5a}$$

subject to

$$0 \le \alpha_i \le p_i^a, \quad i \in N_a \tag{5.5b}$$

$$\alpha_j = 0, \quad j \notin N_a \tag{5.5c}$$

$$(5.4) \tag{5.5d}$$

The objective (5.5a) is the curtailment profit defined in (5.3). Constraints (5.5b) and (5.5c) indicate that the aggregator can only curtail generation at its own nodes, and the amount of curtailment cannot exceed the amount of generation available to it. Constraints (5.5d), which are the KKT conditions, enforce the locational marginal pricing adopted by the ISO. Note that if there is a curtailment limit above which, for example, curtailment can be detected by the ISO, one can simply replace p_i^a in (5.5b) by $\min\{p_i^a, \tau_i\}$ to account for it.

An important note about this problem is that we have assumed the aggregator has complete knowledge of the network topology ($\mathbf{G}$) and state estimates ($\mathbf{p}$ and $\mathbf{d}$). This is, perhaps, optimistic; however one would hope that the market design is such that aggregators do not have profitable manipulations even with such knowledge. The results in this chapter indicate that this is not the case.

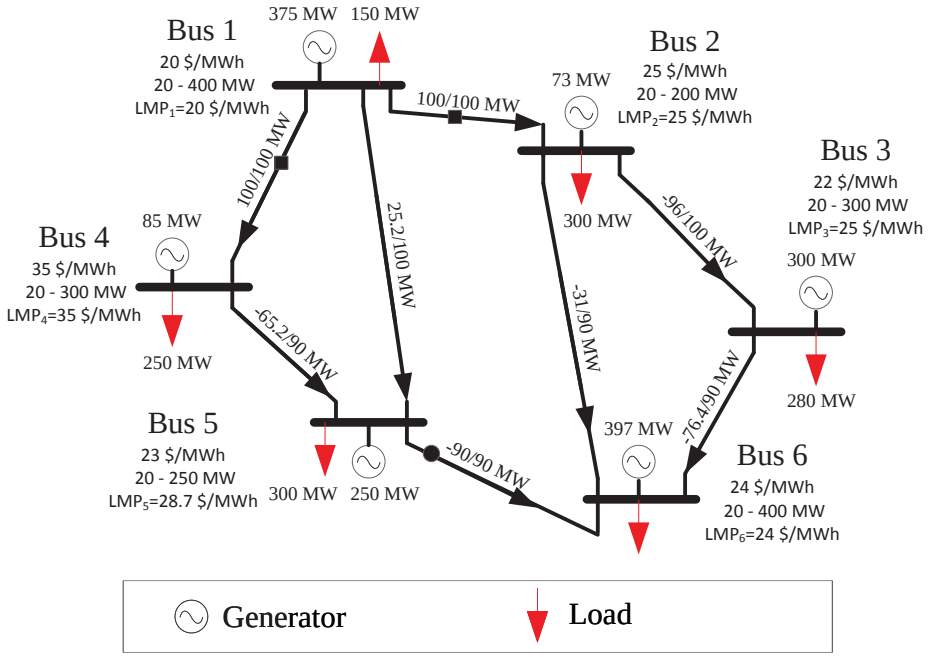


Figure 5.1: The 6-bus example network from [35], used to illustrate the effect of curtailment.

5.4 The Impact of Strategic Curtailment

In this section, we demonstrate the potential impact of strategic curtailment in practical settings. We first provide an illustrative example of how curtailment leads to a larger profit for a simple single-bus aggregator in a small, 6-bus, network. Then we show the effect of strategic curtailment in more realistic settings, using IEEE 14-, 30-, and 57-bus test cases and their enhanced versions from NICTA Energy System Test case Archive [60].

5.4.1 An Illustrative Example

Fig. 5.1 shows a 6-bus example network from [35], in which the amounts of generation are 375.20, 73.00, 299.60, 84.80, 250.00, and 397.40 MW. The loads and the original offer prices for the generators are shown in the figure. At the normal conditions, the lines l_{12}, l_{14} and l_{56} are carrying their maximum flow, and the real-time LMPs are 20.0, 25.0, 25.0, 35.0, 28.7, and 24.0 $\$/MWh$, respectively.

Assume that the aggregator owns node 1 and aims to increase its profit by curtailing the generation at this node. It can be seen that by curtailing just 0.15 MW generation at node 1 (i.e., from 375.20 MW to 375.05 MW), the binding/non-binding constraints in problem (5.1) change, and as a result, the ISO will determine the new LMPs as 25.8,

25.0, 25.0, 35.0, 30.6, and 24.0 $\$/MWh$. Fig. 5.2 shows the LMPs, before and after the curtailment. In this case, the curtailment profit is $\gamma = 25.8 \times 375.05 - 20 \times 375.20 = 2172\ ^{\$}/_{h}$,which means that the aggregator has been able to increase its profit by 2172 $\$/h$ during that dispatch interval.

5.4.2 Case Studies

We simulate the behavior of aggregators with different sizes, i.e., different number of buses, in a number of different networks. We use the IEEE 14-, 30-, and 57-bus test cases. Since studying market manipulation makes sense only when there is congestion in the network, we scale the demand (or equivalently the line flow limits) until there is some congestion in the network. In order to examine the profit and market power of aggregator as a function of its size, we assume that the way the aggregator grows is by sequentially adding random buses to its set (more or less like the way, for example, a solar firm grows). Then, at any fixed set of buses, it can choose different curtailment strategies to maximize its profit. In other words, for each of its nodes, it should decide whether to curtail or not (assuming that the amount of curtailment has been fixed to a small portion). We assume that the total generation of the aggregator in each bus is 10 MW, and it is able to curtail 1% of it (0.1 MW).

For each of the three networks, Fig. 5.3 shows the profit for a random sequence of nodes. Comparing the no-curtailment profit with the strategic-curtailment profit reveals an interesting phenomenon. As the size of the aggregator (number of its buses) grows, not only does the profit increase (which is expected), but also the

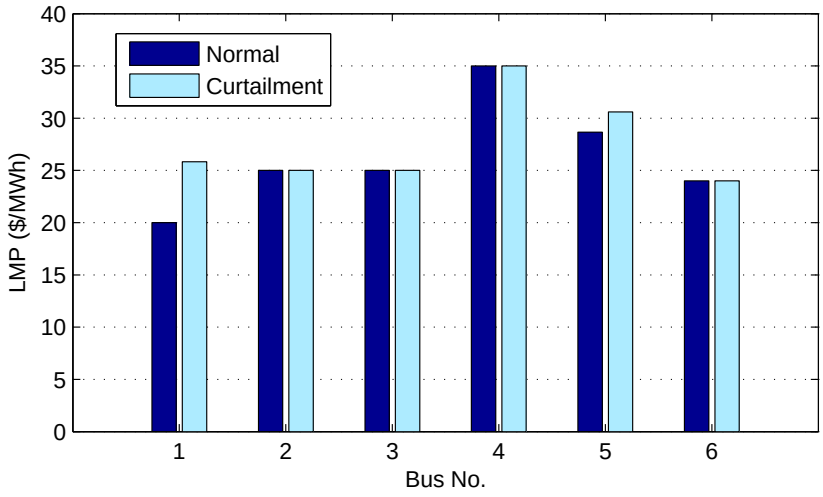


Figure 5.2: The locational marginal prices for the 6-bus example before and after the curtailment.

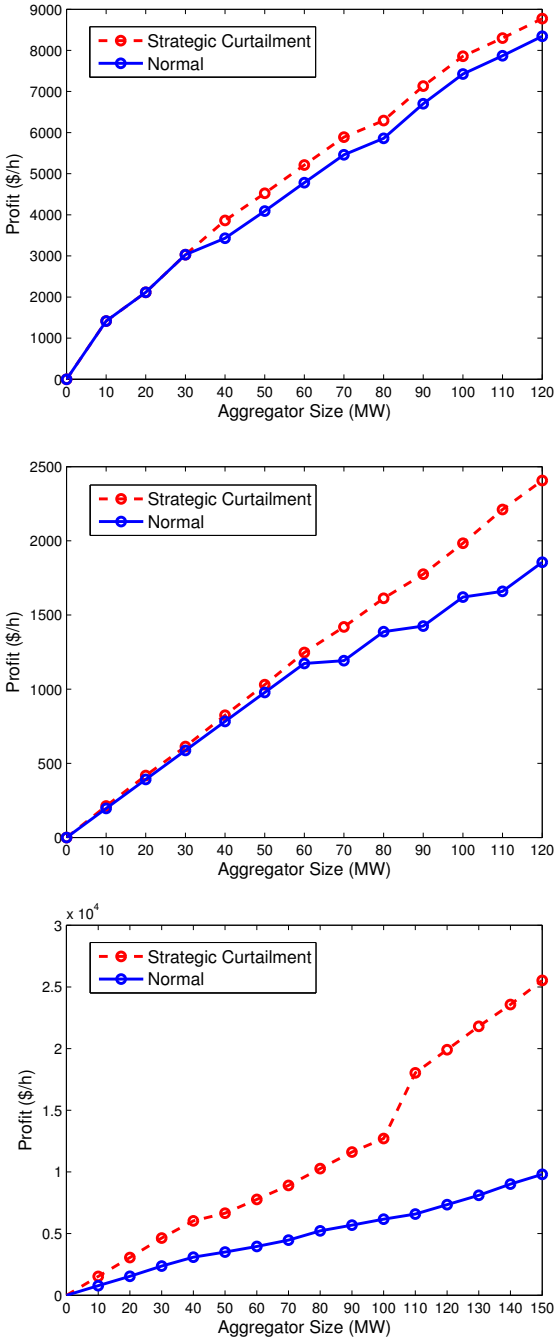


Figure 5.3: The profit under the normal (no-curtailment) condition and under (optimal) strategic curtailment, as a function of size of the aggregator in IEEE test case networks: a) IEEE 14-Bus Case, b) IEEE 30-Bus Case, and c) IEEE 57-Bus Case. The difference between the two curves is the curtailment profit.

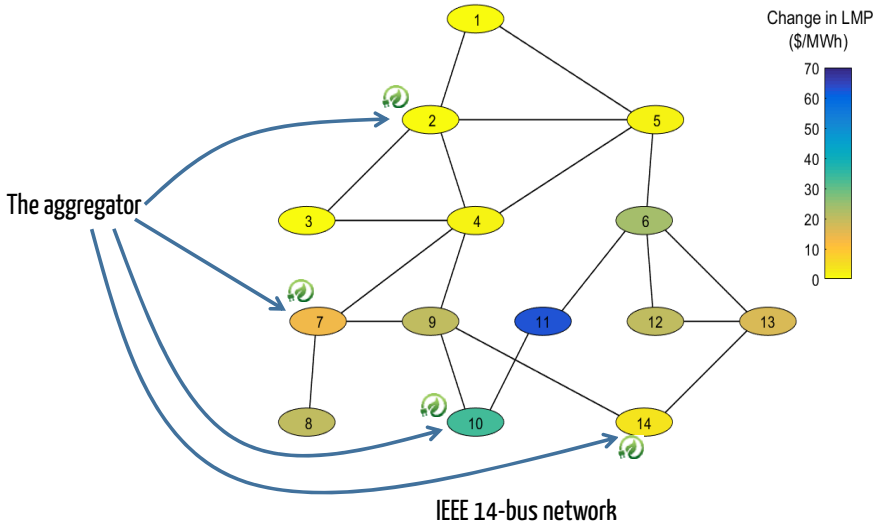


Figure 5.4: A heat map of the impact of coordinated curtailment on the prices in the IEEE 14-bus network. Aggregator nodes are 2, 7, 10, and 14.

difference between the two curves increases, which is the "curtailment profit." More specifically, the latter does not need to happen in theory. However in practice, it is observed most of the time, and it highlights that larger aggregators have higher incentive to behave strategically, and they can indeed gain more from curtailment.

The other important question is what the impact of strategic curtailment on the price of each bus of the network (not necessarily just the aggregator's buses) is. This is important in many scenarios, like the effect of such coordinated manipulations on consumers or the effect of competing firms on each other. Fig. 5.4 shows a heat map of an aggregator's impact on the prices in the IEEE 14-bus network. As one can see, the price of other buses can often be highly impacted as well.

5.5 Optimizing Curtailment Profit

The aggregator's profit maximization problem is challenging to analyze, as one would expect given its bilevel form. In fact, bilevel linear programming is NP-hard to approximate up to any constant multiplicative factor in general [61]. Furthermore, the objective of the program (5.5) is quadratic (bilinear) in the variables, rather than linear. This combination of difficulties means that we cannot hope to provide a complete analytic characterization of the behavior of a profit-maximizing aggregator.

In this section, we begin with the case of a single-bus aggregator and build to the case of general multi-bus aggregators in acyclic networks. For the single-bus

aggregator, the optimal curtailment can be found exactly, in polynomial time. For the general case, we cannot provide an exact algorithm, but we do provide a practical approximation algorithm for general multi-bus aggregators in acyclic networks (e.g., distribution networks).

5.5.1 An Exact Algorithm for Single-Node Aggregators in Arbitrary Networks

Even in the simplest case, when the aggregator has only a single node, i.e., its entire generation is located in a single bus, it is not trivial how to solve the aggregator's profit maximization problem.

The first step toward solving the problem is already difficult. In particular, in order to understand the effect of curtailment on the profit, we first need to understand how curtailment impacts the prices—an impact which is not monotonic in general. Although LMPs are not monotonic in general, it turns out that in single-bus curtailment, the LMP is indeed monotonic with respect to the curtailment. The proof of the following lemma is in Appendix 5.B.

Lemma 25. *The LMP of any bus i is monotonically increasing with respect to the curtailment at that bus. That is,*

$$\lambda_i(\boldsymbol{\alpha}') \geq \lambda_i(\boldsymbol{\alpha})$$

if $\alpha'_i > \alpha_i$, and $\alpha'_j = \alpha_j$ for all $j = [n] \backslash \{i\}$.

A consequence of the above lemma is that the price λ_i is a monotonically increasing staircase function of α_i, for any bus i, as depicted in Fig. 5.5. As α_i increases, if the binding constraints of (5.1) do not change, the dual variables remain the same, and thus the LMPs remain the same (constant intervals). Once a constraint becomes binding/non-binding, the LMP jumps to the next level.

In Fig. 5.5, the two shaded areas show profit at the normal condition and at the curtailment. The difference between the two areas is the curtailment profit. In particular, if the red area is larger than the blue one, the aggregator is able to earn a positive curtailment profit on bus i. The optimal curtailment α_i^* also happens where the red area is maximized. It should be clear that the optimal curtailment always happens at the verge of a price change, not in the middle of a constant interval (otherwise, it can be increased by curtailing less).

Given the knowledge of the network and state estimates, it is possible to find the jump points (i.e., where the binding constraints change) and evaluate them for profitability.

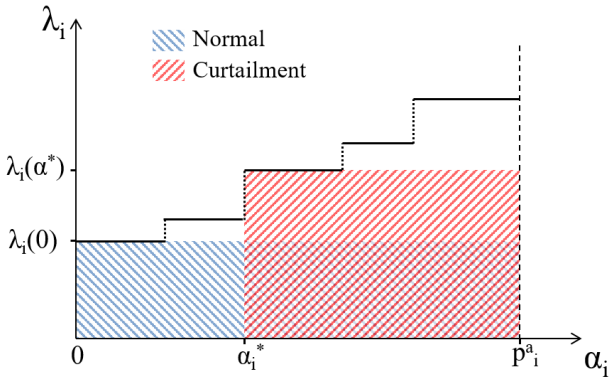


Figure 5.5: The LMP at bus i as a function of curtailed generation at that bus. Shaded areas indicate the aggregator's revenue at the normal condition and at the curtailment.

Therefore, if there are not too many jumps, an exhaustive search over the jump points can yield the optimal curtailment. Based on this observation, we have the following theorem, which is proven in Appendix 5.C.

Theorem 26. *The exact optimal curtailment for an aggregator with a single bus, in an arbitrary network with t lines, can be found by an algorithm with running time $O(t^{3.373})$.*

Clearly, this approach does not extend to large multi-bus aggregators. The following section uses a different and more sophisticated algorithmic approach for that setting.

5.5.2 An Approximation Algorithm for Multi-Bus Aggregators in Radial Networks

In this section, we show that the aggregator profit maximization problem, while hard in general, can be solved in an approximate sense to determine an approximately-feasible approximately-optimal curtailment strategy in polynomial time using an approach based on dynamic programming. In particular, we show that an ϵ-approximation of the optimal curtailment profit can be obtained using an algorithm with running time that is linear in the size of the network and polynomial in $\frac{1}{\epsilon}$.

Before we state the main result of this section, we introduce the notion of an approximate solution to (5.5) in the following definition.

Definition 5. *A solution $(\alpha, f, \lambda^-, \lambda^+, \mu^-, \mu^+, \nu)$ to (5.5) is an **ϵ-accurate solution** if the constraints are violated by at most ϵ and $\gamma(\alpha) \geq \gamma^* - \epsilon$.*

Note that, if one is simply interested in approximating γ^* (as a market regulator would be), the ϵ-constraint violation is of no consequence, and an ϵ-accurate solution of (5.5) suffices to compute an ϵ-approximation to γ^*.

Given the above notion of approximation, our main theorem is as follows (proof in Appendix 5.D):

Theorem 27. *An ϵ-accurate solution to the optimal aggregator curtailment problem* (5.5) *for an n-bus radial network can be found by an algorithm with running time $cn\left(\frac{1}{\epsilon}\right)^9$, where c is a constant that depends on the parameters $p_i^a, B, d, p, \underline{f}, \overline{f}$. On a linear (feeder line) network, the running time reduces to $cn\left(\frac{1}{\epsilon}\right)^6$.*

We now give an informal description of the approximation algorithm. Consider a radial distribution network with nodes labeled $i \in [n]$ (where 1 denotes the substation bus, where the radial network connects to the transmission grid). Radial distribution networks have a *tree* topology (they do not have cycles). We denote bus 1 as the *root* of the tree, and buses with only one neighbor as *leaves*. Every node (except the root) has a unique *parent*, defined as the first node on the unique path connecting it to the root node. The set of nodes k that have a given node i as its parent are said to be its *children*. It can be shown that the strategic curtailment problem on any radial distribution network can be expressed as an equivalent problem on a network where each node has maximum degree 3 (known as a *binary tree*, see Appendix 5.D). Thus, we can limit our attention to networks of this type, where every node has a unique parent and at most 2 children.

For a node i, let $c_1(i), c_2(i)$ denote its children (where $c_1 = \emptyset$, $c_2 = \emptyset$ is allowed since a node can have fewer than two children). We use the shorthand

$$p^{net}(i) = f_{c_1(i)} + f_{c_2(i)} - f_i - (p_i - \alpha_i - d_i).$$

Constraint (5.4a) reduces to $\underline{\Delta p}_i \leq p^{net}(i) \leq \overline{\Delta p_i}$, where $f_1 = 0$ and $f_\emptyset = 0$. The matrix H in (5.4c) is an empty matrix (the nullspace of the matrix B is of dimension 0), so this constraint can be dropped. Using this additional structure, the problem

(5.5) can be rewritten (after some algebra) as:

$$\underset{\lambda,f,\alpha}{\text{maximize}} \qquad \sum_{i=1}^{n} \lambda_i \left(p_i^a - \alpha_i\right) \tag{5.6a}$$

subject to

$$0 \le \alpha_i \le p_i^a, \quad i \in [n] \tag{5.6b}$$

$$\underline{\Delta p}_i \le p^{net}(i) \le \overline{\Delta p}_i, \quad i \in [n] \tag{5.6c}$$

$$\underline{f}_i \le f_i \le \overline{f}_i, \quad i \in [n] \setminus \{1\} \tag{5.6d}$$

$$\lambda_i \quad \begin{cases} \le c_i, & \text{if } p^{net}(i) = \underline{\Delta p}_i \\ = c_i, & \text{if } \underline{\Delta p}_i < p^{net}(i) < \overline{\Delta p}_i \\ \ge c_i, & \text{if } p^{net}(i) = \overline{\Delta p}_i \end{cases}, i \in [n] \tag{5.6e}$$

$$\lambda_{c_j(i)} - \lambda_i \begin{cases} \ge 0, & \text{if } f_i = \underline{f}_i \\ = 0, & \text{if } \underline{f}_i < f_i < \overline{f}_i \\ \le 0, & \text{if } f_i = \overline{f}_i \end{cases}, i \in [n], j = 1, 2 \tag{5.6f}$$

where λ_i is the LMP at bus i. Note that we assumed that there is some aggregator generation and potential curtailment at every bus (however this is not restrictive, since we can simply set $p_i^a = 0$ at buses where the aggregator owns no assets).

Define $x_i = (\lambda_i, f_i, \alpha_i)$, it is easy to see that (5.6) is of the form

$$\begin{aligned} \max_{x} \quad & \sum_{i=1}^{n} g_i(x_i) \\ \text{s.t.} \quad & h_i\left(x_i, x_{c_1(i)}, x_{c_2(i)}\right) \le 0, \quad i \in [n] \end{aligned}$$

for some functions $g_i(.)$ and $h_i(.)$. This form is amenable to dynamic programming, since, if we fix the value of x_i, the optimization problem for the subtree under i is decoupled from the rest of the network. Set $\kappa_n(x) = 0$, define κ_i for $i < n$ recursively as

$$\kappa_i(x) = \max_{\substack{x_{c_1(i)}, x_{c_2(i)} \\ h_i\left(x, x_{c_1(i)}, x_{c_2(i)}\right) \le 0}} \sum_{j=1}^{2} g_{c_j(i)}\left(x_{c_j(i)}\right) + \kappa_{c_j(i)}\left(x_{c_j(i)}\right).$$

Then, the optimal value can be computed as $\gamma^* = \max_x \kappa_1(x) + g_1(x)$. However, the above recursion requires an infinite-dimensional computation at every step, since the value of κ_i needs to be calculated for *every* value of x. To get around this, we note that the variables λ_i, f_i, α_i are bounded, and hence x_i can be discretized to lie in a

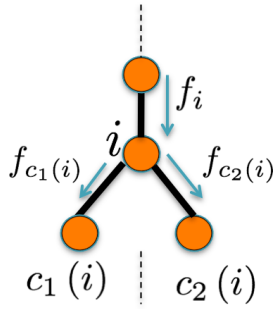


Figure 5.6: The representation of a binary tree. For any node i, and its children denoted $c_1(i), c_2(i)$.

certain set $\mathcal{X}_i$ such that every feasible x_i is at most $\delta(\epsilon_i)$ away (in infinity-norm sense) from some point in $\mathcal{X}_i$ (Lemma 29). The discretization error can be quantified, and this error bound can be used to relax the constraint to $h_i\,(x_i, x_{i+1}) \leq \epsilon$, guaranteeing that any solution to (5.5) is feasible for the relaxed constraint. This allows us to define a dynamic program (Algorithm 3).

Algorithm 3 Dynamic programming on binary tree

$S \leftarrow \{i : c_1\,(i) = \emptyset, c_2\,(i) = \emptyset\}$
$\kappa_i\,(x) \leftarrow 0\ \forall x \in \mathcal{X}_i, i \in S$
while $|S| \leq n$ **do**
  $S' \leftarrow \{i \notin S : c_1\,(i)\,, c_2\,(i) \in S\}$
  $\forall i \in S', \forall x \in \mathcal{X}_i$:

$$\kappa_i\,(x) \leftarrow \max_{\substack{x_1' \in \mathcal{X}_{c_1(i)}, x_2' \in \mathcal{X}_{c_2(i)} \\ h_i\left(x, x_1', x_2'\right) \leq \epsilon}} \sum_{j=1,2} g_{c_j(i)}\left(x_j'\right) + \kappa_{c_j(i)}\,(x')$$

  $S \leftarrow S \cup S'$
end while
$\gamma \leftarrow \max_{x \in \mathcal{X}_1} \kappa_1\,(x) + g_1\,(x)$

The algorithm essentially starts at the leaves of the tree and proceeds towards the root, at each stage updating κ for nodes whose children have already been updated (stopping at root). Along with the discretization error analysis in Appendix 5.D, this essentially concludes Theorem 27.

It is worth noting that previous work on distribution level markets have used AC power flow models (at least in some approximate form) due to the importance of voltage constraints and reactive power in a distribution system [158]. Our approach

extends in a straightforward way to this setting as well, as the dynamic programming structure remains preserved (the KKT conditions will simply be replaced by the corresponding conditions for the AC-based market clearing mechanism).

5.5.3 Evaluation of the Approximation Algorithm

To evaluate the performance of our approximation algorithm on acyclic networks, we run it on a number of small test networks and compare the results with the brute-force optimal values. The algorithm indeed finds solutions within the prespecified error range (and often exact) in reasonable time.

As an example, for an acyclic version of the IEEE 9-bus network (taken from [116]), we demonstrate the suboptimality gap of the solution versus the running time in Fig. 5.8. At each point of the graph, the error percentage (y-axis) is bounded by a constant factor of ϵ. Clearly, the smaller ϵ we choose, the longer the running time is, but the smaller the error becomes. As one can see, the error drops pretty quickly.

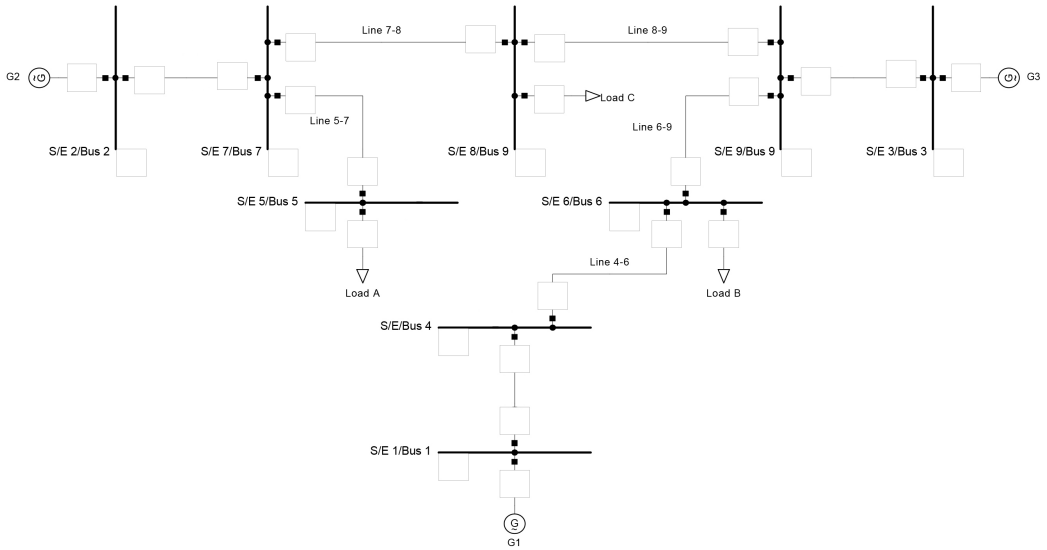


Figure 5.7: The 9-bus acyclic network from [116], used for the evaluation of the proposed approximation algorithm.

We should remark that the network chosen here was small in order to allow for comparison with the optimal value. However, the main advantage of our algorithm is that it is scalable, while the brute-force becomes intractable quickly.

5.6 Concluding Remarks

Understanding the potential for market manipulation by aggregators is crucial for electricity market efficiency in the new era of renewable energy. In this chapter, we

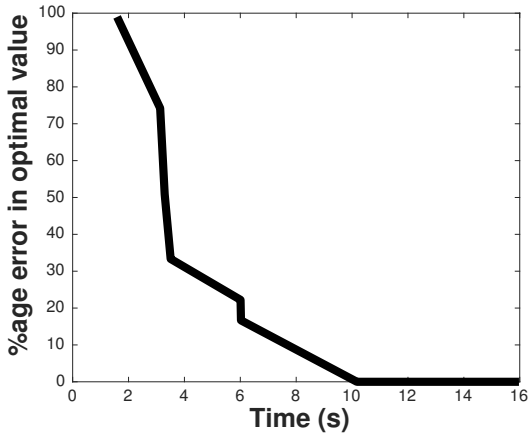


Figure 5.8: The difference from the optimal solution as a function of the running time of the algorithm, in the 9-bus network with 1% curtailment allowance.

characterized the profit an aggregator can make by strategically curtailing generation in the ex-post market as the outcome of a bi-level optimization problem. This model captures the realistic price clearing mechanism in the electricity market. We showed through simulations on realistic test cases that there is potentially large profit for aggregators by manipulating the LMPs in the electricity market. When the aggregator is located in a single bus, we have shown that the locational marginal price is monotonically increasing with the curtailment, and we have an exact polynomial-time algorithm to solve the aggregators profit maximization problem.

The aggregator's strategic curtailment problem in a general setting is a difficult bi-level optimization problem, which is intractable. However, we showed that for radial distribution networks (where aggregators are likely located), there is an efficient algorithm to approximate the solution up to arbitrary precision. We also demonstrated via simulation on a distribution test case that our algorithm can efficiently find the approximately optimal curtailment strategy.

We view this work as a first step in understanding market power of aggregators, and more generally, towards market design for integrating renewable energy and demand response from geographically distributed sources. With the result of this work, it is interesting to ask what the operator can do to address this problem, and in particular, how to design market rules for aggregators to maximize the contribution of renewable energy yet mitigate the exercise of market power. Also, extending the analysis to the case of multiple aggregators in the market is another interesting direction for future research.

5.A Connections between Curtailment Profit and Market Power

As mentioned earlier, there has been significant work on market power in electricity markets, but work is only beginning to emerge on the market power of renewable generation producers. One important work from this literature is [221], and the following is the proposed notion of market power from that work.

Definition 6. *For $\alpha_i^* \geq 0$, the **market power** (ability) of the aggregator is defined as*

$$\eta_i = \left(\frac{\lambda_i(\alpha^*) - \lambda_i(0)}{\lambda_i(0)}\right) / \left(\frac{\alpha_i^*}{p_i^a}\right) \tag{5.7}$$

In this definition, the value of η_i captures the ability of the generator/aggregator to exercise market power. Intuitively, in a market with high value of η_i, the aggregator can significantly increase the price by curtailing a small amount of generation.

Interestingly, the optimal curtailment profit is closely related to this notion of market power. We summarize the relationship in the following proposition.

Proposition 28. *If the curtailment profit γ is positive, then the market power $\eta_i > 1$. Furthermore, the larger the curtailment profit is, the higher the market power.*

Proof. From the definition of $\gamma(\alpha^*) = \lambda_i(\alpha^*)(p_i^a - \alpha_i^*) - \lambda_i(0)p_i^a$, it follows that

$$\begin{aligned}
\frac{\gamma(\alpha^*)}{\lambda_i(0)(p_i^a - \alpha_i^*)} &= \frac{\lambda_i(\alpha^*)}{\lambda_i(0)} - \frac{p_i^a}{p_i^a - \alpha_i^*} \\
&= 1 + \frac{\lambda_i(\alpha^*) - \lambda_i(0)}{\lambda_i(0)} - (1 - \frac{\alpha_i^*}{p_i^a})^{-1} \\
&\simeq 1 + \frac{\lambda_i(\alpha^*) - \lambda_i(0)}{\lambda_i(0)} - (1 + \frac{\alpha_i^*}{p_i^a}) \\
&= \frac{\lambda_i(\alpha^*) - \lambda_i(0)}{\lambda_i(0)} - \frac{\alpha_i^*}{p_i^a}.
\end{aligned} \tag{5.8}$$

Therefore we have

$$\begin{aligned}
\frac{p_i^a}{\lambda_i(0)(p_i^a - \alpha_i^*)\alpha_i^*}\gamma(\alpha^*) &= \left(\frac{\lambda_i(\alpha^*) - \lambda_i(0)}{\lambda_i(0)}\right) / \left(\frac{\alpha_i^*}{p_i^a}\right) - 1 \\
&= \eta_i - 1.
\end{aligned}$$

Since the left-hand side parameters are all positive, if $\gamma(\alpha^*) > 0$, we can conclude that $\eta_i > 1$. Moreover, it is clear that the larger the value of $\gamma(\alpha^*)$ is, the higher the value of η_i is. Note that we used the approximation $(1 - \frac{\alpha_i^*}{p_i^a})^{-1} \simeq 1 + \frac{\alpha_i^*}{p_i^a}$, since

the curtailment is small with respect to the generation; however, the right-hand side expression (5.8) is an upper bound on the left-hand side anyway, and the result holds exactly. □

This proposition highlights that the notion of market power in [221] is consistent with an aggregator seeking to maximize its curtailment profit, and higher curtailment profit corresponds to more market power.

5.B Proof of Lemma 25 (Monotonicity of LMP)

Let us take a look at the ISO's optimization problem (5.1), which is a linear program. It is not hard to see that the dual of this problem is as follows.

$$\begin{aligned}
\underset{\boldsymbol{\lambda}^-,\boldsymbol{\lambda}^+,\boldsymbol{\mu}^-,\boldsymbol{\mu}^+,\boldsymbol{\nu}}{\text{maximize}} \quad & (\underline{\mathbf{\Delta p}} + \mathbf{p} - \boldsymbol{\alpha} - \mathbf{d})^T \boldsymbol{\lambda}^- + \\
& (-\mathbf{p} + \boldsymbol{\alpha} + \mathbf{d} - \overline{\mathbf{\Delta p}})^T \boldsymbol{\lambda}^+ + \underline{\mathbf{f}}^T \boldsymbol{\mu}^- - \overline{\mathbf{f}}^T \boldsymbol{\mu}^+ && \text{(5.9a)} \\
\text{subject to} & \\
& \mathbf{B}^T(\boldsymbol{c} + \boldsymbol{\lambda}^+ - \boldsymbol{\lambda}^-) - \boldsymbol{\mu}^- + \boldsymbol{\mu}^+ + \mathbf{H}^T \boldsymbol{\nu} = \mathbf{0} && \text{(5.9b)} \\
& \boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+, \boldsymbol{\mu}^-, \boldsymbol{\mu}^+ \geq 0 && \text{(5.9c)}
\end{aligned}$$

If one focuses on the terms involving α_i for a certain i, the objective of the above optimization problem is in the form: $(\underline{\Delta p}_i + p_i - \alpha_i - d_i)\lambda_i^- + (-p_i + \alpha_i + d_i - \overline{\Delta p}_i)\lambda_i^+$ plus a linear function of the rest of the variables (i.e., the rest of $\boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+$, as well as $\boldsymbol{\mu}^-, \boldsymbol{\mu}^+, \boldsymbol{\nu}$). There is no $\boldsymbol{\alpha}$ in the constraints, and the first two terms of this objective are the only parts where α_i appears (and with opposite signs).

We need to show that if α_i is changed to $\alpha_i + \delta$ for some $\delta > 0$, then $c_i + \lambda_i^{+new} - \lambda_i^{-new} \geq c_i + \lambda_i^+ - \lambda_i^-$, where $\lambda_i^{+new}, \lambda_i^{-new}$ are the optimal solutions of the new problem.

We prove this in a general setting. Consider the following two optimization problems.

$$\begin{aligned}
f^* = \sup_{\substack{x_1,x_2\in\mathbb{R} \\ x_3\in\mathbb{R}^m}} \quad & a_1x_1 + a_2x_2 + a_3^Tx_3 && \text{(5.10a)} \\
\text{s.t.} \quad & (x_1, x_2, x_3) \in S && \text{(5.10b)}
\end{aligned}$$

$$\begin{aligned}
f^{*new} = \sup_{\substack{x_1,x_2\in\mathbb{R} \\ x_3\in\mathbb{R}^m}} \quad & (a_1 - \delta)x_1 + (a_2 + \delta)x_2 + a_3^Tx_3 && \text{(5.11a)} \\
\text{s.t.} \quad & (x_1, x_2, x_3) \in S && \text{(5.11b)}
\end{aligned}$$

Assume that the optimal values of the problems are attained at (x_1^*, x_2^*, x_3^*) and $(x_1^{*new}, x_2^{*new}, x_3^{*new})$, respectively.

We claim that $x_2^{*new} - x_1^{*new} \geq x_2^* - x_1^*$. (This precisely implies the LMP condition in our case, i.e., $\lambda_i^{+new} - \lambda_i^{-new} \geq \lambda_i^+ - \lambda_i^-$).

Suppose by way of contradiction that $x_2^{*new} - x_1^{*new} < x_2^* - x_1^*$.
We know that $a_1x_1^* + a_2x_2^* + a_3^Tx_3^* \geq a_1x_1 + a_2x_2 + a_3^Tx_3,\ \forall (x_1, x_2, x_3) \in S$.

Therefore we have

$$
\begin{aligned}
&(a_1 - \delta)x_1^{*new} + (a_2 + \delta)x_2^{*new} + a_3^T x_3^{*new} \\
&= a_1 x_1^{*new} + a_2 x_2^{*new} + a_3^T x_3^{*new} - \delta x_1^{*new} + \delta x_2^{*new} \\
&\leq a_1 x_1^* + a_2 x_2^* + a_3^T x_3^* + \delta(x_2^{*new} - x_1^{*new}) \\
&< a_1 x_1^* + a_2 x_2^* + a_3^T x_3^* + \delta(x_2^* - x_1^*) \\
&= (a_1 - \delta)x_1^* + (a_2 + \delta)x_2^* + a_3^T x_3^*.
\end{aligned}
$$

The first inequality above follows from the fact that $(x_1^{*new}, x_2^{*new}, x_3^{*new}) \in S$. Now the above implies that $(x_1^{*new}, x_2^{*new}, x_3^{*new})$ is not the optimal solution of (5.11), and it is a contradiction.

As a result, $x_2^{*new} - x_1^{*new} \geq x_2^* - x_1^*$. □

5.C Proof of Theorem 26 (Exact Single-Bus)

Since we are in the single-bus curtailment regime, α has only one nonzero component. For the sake of convenience, we denote that element itself by a scalar α throughout this proof (no α is vector in this proof). The proof consists of the following two pieces: 1) From each jump point, the point where the next jump happens can be computed in polynomial time and 2) There are at most polynomially (in this case even linearly) many jumps.

Assuming that the solution to the program (5.1) is unique, for any fixed value of α, exactly t of the constraints (5.1b), (5.1c), and (5.1d) are binding (active). We can express these binding constraints as

$$Af = b(\alpha),$$

where $A \in \mathbb{R}^{t\times t}$ is an invertible matrix, and $b(\alpha) \in \mathbb{R}^t$ is a vector that depends on α. As long as the binding constraints do not change, the matrix A is fixed, and the optimal solution is linear in α (i.e., $f = A^{-1}b(\alpha)$). Then, for simplicity, we can express the solution as $f(\alpha) = f_0 + \alpha a$, for some t-vectors f_0 and a.

Now, if we look at the non-binding (inactive) constraints of (5.1), they can also be expressed as

$$\tilde{A}f < \tilde{b},$$

for some matrix $\tilde{A}$ and vector $\tilde{b}$ of appropriate dimensions. Inserting f into this set of inequalities yields $\tilde{A}f_0 + \alpha\tilde{A}a < \tilde{b}$, or equivalently

$$\alpha(\tilde{A}a)_i < \tilde{b}_i - (\tilde{A}f_0)_i,$$

for all $i = 1, 2, \ldots, (2n + 2t - rank(G))$.

Now we need to figure out that, with increasing α, which of the non-binding constraints becomes binding first and with exactly how much of an increase in α. If for some i we have $(\tilde{A}a)_i \leq 0$, then it is clear that increasing α cannot make constraint i binding. If $(\tilde{A}a)_i > 0$ then the constraint can be written as

$$\alpha < \frac{\tilde{b}_i - (\tilde{A}f_0)_i}{(\tilde{A}a)_i}.$$

Computing the right-hand side for all i, and taking their minimum, tells us exactly which constraint will become binding next and how much change in the current value of α results in that.

The complexity of this procedure is $O(t^{2.373})$ for computing f_0 and a, plus $O(t(2n + 2t)) = O(nt + t^2)$ for computing the lowest bound among all the constraints. Hence the complexity is $O(t^{2.373})$.

The above procedure describes how the next jump point can be computed efficiently from the current point. The exact same procedure can be repeated for reaching the subsequent jump points. All that remains is to show the second piece of the proof, which is that the number of jump points are bounded polynomially. To show the last part, note that by increasing α, if a binding constraint becomes non-binding, it will not become binding again. As a result, each constraint can change at most twice, and therefore, the number of jumps is at most twice the number of constraints. Thus, the number of jumps is $O(n + t)$, and the overall complexity of the algorithm is $O((n + t)t^{2.373}) = O(t^{3.373})$. □

5.D Proof of Theorem 27 (Approximate Multi-Bus)

Lemma 29 (δ-discretization)**.** *Given a set $C \subset [\underline{L_1}, \overline{L_1}] \times \cdots \times [\underline{L_k}, \overline{L_k}]$, there exists a finite set $\mathcal{X}$ such that*

$$\forall z \in C \quad \exists z' \in \mathcal{X}, \max_{1 \le i \le k} |z_i - z'_i| \le \delta$$

and $\mathcal{X}$ contains at most V/δ^k points, where $V = \prod_{i=1}^{k} (\overline{L_i} - \underline{L_i})$ is a constant (the volume of the box). $\mathcal{X}$ is said to be an δ-discretization of C and written as $\mathcal{X}$ (δ).

Lemma 30 (Reduction to Binary Tree)**.** *Any tree with arbitrary degrees can be reduced to a binary tree by introducing additional dummy nodes to the network.*

Proof. Take any node b in the tree with some parent a and $k > 2$ children $c_1, \ldots, c_k$. There exists $m > 0$ such that $2^m < k \le 2^{m+1}$ for some m. We will show that this subgraph can be made a binary tree by introducing $O(k)$ dummy nodes (in m levels) between b and its children. The additional nodes and edges are defined as follows:

$$b \to b_0,\ b \to b_1,$$

$$b_0 \to b_{00},\ b_0 \to b_{01},\ b_1 \to b_{10},\ b_1 \to b_{11},$$

$$b_{00} \to b_{000},\ b_{00} \to b_{001},\ \ldots,\ b_{11} \to b_{111},$$

up to m levels:

$$b_{0\ldots00} \to c_1,\ b_{0\ldots00} \to c_2,\ b_{0\ldots01} \to c_3\ \ldots$$

This is transparently a binary tree with $O(k)$ nodes. Each of the new nodes has zero injection, and effectively the incoming flow from its parent is just split in some way between its children. This in fact enforces the flow conservation constraint at b. Similar construction can be applied to any node of the tree with more than two children, until no such node exists. It can be seen that the number of nodes in the new graph is still linear in n. □

So any tree can be transformed to a binary one by the above procedure. For the rest of the analysis, we focus on the ϵ-approximation of the dynamic program on the resulting binary tree. The optimization problem (5.5) on a binary tree, can be written after some algebra as the following.

$$\max_{\boldsymbol{\lambda}, \mathbf{f}, \alpha} \quad \sum_{i=1}^{n} \lambda_i (p_i^a - \alpha_i) \tag{5.12a}$$

subject to

$$0 \leq \alpha_i \leq p_i^a, \quad i = 1, \ldots, n \tag{5.12b}$$

$$\underline{\Delta p}_i \leq f_{c_1(i)} + f_{c_2(i)} - f_i - p_i + \alpha_i + d_i \leq \overline{\Delta p}_i,$$
$$i = 1, \ldots, n \tag{5.12c}$$

$$\underline{f}_i \leq f_i \leq \overline{f}_i, \quad i = 2, \ldots, n \tag{5.12d}$$

$$\begin{cases} (\lambda_i - c_i)(f_{c_1(i)} + f_{c_2(i)} - f_i - p_i + \alpha_i + d_i - \underline{\Delta p}_i) \geq 0 \\ (\lambda_i - c_i)(f_{c_1(i)} + f_{c_2(i)} - f_i - p_i + \alpha_i + d_i - \overline{\Delta p}_i) \geq 0 \end{cases},$$
$$i = 1, \ldots, n \tag{5.12e}$$

$$\begin{cases} (\lambda_i - \lambda_{c_j(i)})(\underline{f}_{c_j(i)} - f_{c_j(i)}) \geq 0 \\ (\lambda_i - \lambda_{c_j(i)})(\overline{f}_{c_j(i)} - f_{c_j(i)}) \geq 0 \end{cases},$$
$$i = 1, \ldots, n, \quad j = 1, 2 \tag{5.12f}$$

The constraints $0 \leq \alpha_i \leq p_i^a$ and $\underline{f}_i \leq f_i \leq \overline{f}_i$, along with a prior bound on lambda $\underline{\lambda} \leq \lambda \leq \overline{\lambda}$ can be used to define the box where $x_i = (\lambda_i, f_i, \alpha_i)$ lives. Then, an ϵ-accurate solution is a solution to the following problem.

$$\max_{\boldsymbol{\lambda}, \mathbf{f}, \alpha} \quad \sum_{i=1}^{n} \lambda_i (p_i - \alpha_i) \tag{5.13a}$$

subject to

$$\underline{\Delta p}_i - \epsilon \leq f_{c_1(i)} + f_{c_2(i)} - f_i - p_i + \alpha_i + d_i \leq \overline{\Delta p}_i + \epsilon,$$
$$i = 1, \ldots, n \tag{5.13b}$$

$$\begin{cases} (\lambda_i - c_i)(f_{c_1(i)} + f_{c_2(i)} - f_i - p_i + \alpha_i + d_i - \underline{\Delta p}_i) \geq -\epsilon \\ (\lambda_i - c_i)(f_{c_1(i)} + f_{c_2(i)} - f_i - p_i + \alpha_i + d_i - \overline{\Delta p}_i) \geq -\epsilon \end{cases},$$
$$i = 1, \ldots, n \tag{5.13c}$$

$$\begin{cases} (\lambda_i - \lambda_{c_j(i)})(\underline{f}_{c_j(i)} - f_{c_j(i)}) \geq -\epsilon \\ (\lambda_i - \lambda_{c_j(i)})(\overline{f}_{c_j(i)} - f_{c_j(i)}) \geq -\epsilon \end{cases},$$
$$i = 1, \ldots, n, \quad j = 1, 2 \tag{5.13d}$$

Assuming a δ-discretization of the constraint set, each of the constraints (as well as ϵ-accuracy of the objective) imposes a bound on the value of δ. For example, constraint (5.13c) requires $4\delta \leq \epsilon$. (Note that we could have defined different deltas $\delta^{\lambda}, \delta^{f}, \delta^{\alpha}$ for different variables, and in that case, we would have had $3\delta^{f} + \delta^{\alpha} \leq \epsilon$, but for simplicity, we took all the deltas to be the same.) Similar bounds on δ can be obtained from the other constraints, and taking the lowest upper bound implies the existence of a constant c' (that depends on the parameters) such that $\delta \leq \epsilon/c'$.

As a result, we have a δ-discretization with $|\mathcal{X}| = V/\delta^3 = c'^3 V/\epsilon^3$ number of points, for any node. Therefore, the computational complexity over any node will be $|\mathcal{X}|^3$, because we have $|\mathcal{X}|$ many values for the node itself and $|\mathcal{X}|$ many values for any of its two children. Since there are n nodes, the overall complexity of the algorithm will simply be $n|\mathcal{X}|^3 = nc'^9 V^3/\epsilon^9 = cn/\epsilon^9$. □

Part III

Distributed Computation

Chapter 6

DISTRIBUTED SOLUTION OF LARGE-SCALE SYSTEMS OF EQUATIONS

[1] Navid Azizan et al. "Distributed Solution of Large-Scale Linear Systems via Accelerated Projection-Based Consensus". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 6358–6362. DOI: 10.1109/ICASSP.2018.8462630.

[2] Navid Azizan et al. "Distributed Solution of Large-Scale Linear Systems via Accelerated Projection-Based Consensus". In: *IEEE Transactions on Signal Processing* 67.14 (2019), pp. 3806–3817. DOI: 10.1109/TSP.2019.2917855.

Solving a large-scale system of linear equations is a key step at the heart of many algorithms in scientific computing, machine learning, and beyond. When the problem dimension is large, computational and/or memory constraints make it desirable, or even necessary, to perform the task in a distributed fashion. In this chapter, we consider a common scenario in which a taskmaster intends to solve a large-scale system of linear equations by distributing subsets of the equations among a number of computing machines/cores. We propose a new algorithm called *Accelerated Projection-based Consensus (APC)*, in which, at each iteration, every machine updates its solution by adding a scaled version of the projection of an error signal onto the nullspace of its system of equations, and the taskmaster conducts an averaging over the solutions with momentum. The convergence behavior of the proposed algorithm is analyzed in detail and analytically shown to compare favorably with the convergence rate of alternative distributed methods, namely distributed gradient descent, distributed versions of Nesterov's accelerated gradient descent and heavy-ball method, the block Cimmino method, and ADMM. On randomly chosen linear systems, as well as on real-world data sets, the proposed method offers significant speed-up relative to all the aforementioned methods. Finally, our analysis suggests a novel variation of the distributed heavy-ball method, which employs a particular distributed preconditioning, and which achieves the same theoretical convergence rate as the proposed consensus-based method.

6.1 Introduction

With the advent of big data, many analytical tasks of interest rely on distributed computations over multiple processing cores or machines. This is either due to the inherent complexity of the problem, in terms of computation and/or memory, or due to the nature of the data sets themselves that may already be dispersed across machines. Most algorithms in the literature have been designed to run in a sequential fashion, as a result of which, in many cases, their distributed counterparts have yet to be devised. In order to devise efficient distributed algorithms, one has to address a number of key questions, such as: (a) What computation should each worker carry out, (b) What is the communication architecture, and what messages should be communicated between the processors, (c) How does the distributed implementation fare in terms of computational complexity, and (d) What is the rate of convergence in the case of iterative algorithms.

In this chapter, we focus on solving a large-scale system of linear equations, which is one of the most fundamental problems in numerical computation, and lies at the heart of many algorithms in engineering and the sciences. In particular, we consider the setting in which a taskmaster intends to solve a large-scale system of equations in a distributed way with the help of a set of computing machines/cores (Figure 6.1). This is a common setting in many computing applications, and the task is mainly distributed because of high computational and/or memory requirements (rather than physical location as in sensor networks).

This problem can in general be cast as an optimization problem, with a cost function that is separable in the data[1] (but not in the variables). Hence, there are general approaches to construct distributed algorithms for this problem, such as distributed versions of gradient descent [228, 173, 222] and its variants (e.g., Nesterov's accelerated gradient [151] and heavy-ball method [169]), as well as the so-called Alternating Direction Method of Multipliers (ADMM) [44] and its variants. ADMM has been widely used [100, 62, 225] for solving various convex optimization problems in a distributed way, and in particular for consensus optimization [144, 185, 139], which is the relevant one for the type of separation that we have here. In addition to the optimization-based methods, there are a few distributed algorithms designed specifically for solving systems of linear equations. The most famous one of these is what is known as the block Cimmino method [69, 191, 11], which is a block

[1] Solving a system of linear equations, $Ax = b$, can be set up as the optimization problem $\min_x \|Ax - b\|^2 = \min_x \sum_i \|(Ax)_i - b_i\|^2$.

row-projection method [45], and is in a way a distributed implementation of the Kaczmarz method [111]. Another algorithm has been recently proposed in [134, 146], where a consensus-based scheme is used to solve a system of linear equations over a network of autonomous agents. Our algorithm bears some resemblance to all of these methods, but as it will be explained in detail, it has much faster convergence than any of them.

Our main contribution is the design and analysis of a new algorithm for distributed solution of large-scale systems of linear equations, which is significantly faster than all the existing methods. In our methodology, the taskmaster assigns a subset of equations to each of the machines and invokes a distributed consensus-based algorithm to obtain the solution to the original problem in an iterative manner. At each iteration, each machine updates its solution by adding a scaled version of the projection of an error signal onto the nullspace of its system of equations, and the taskmaster conducts an averaging over the solutions with momentum. The incorporation of a momentum term in both projection and averaging steps results in accelerated convergence of our method, compared to the other projection-based methods. For this reason, we refer to this method as *Accelerated Projection-based Consensus (APC)*. We provide a complete analysis of the convergence rate of APC (Section 6.3), as well as a detailed comparison with all the other distributed methods mentioned above (Section 6.4). Also, by empirical evaluations over both randomly chosen linear systems and real-world data sets, we demonstrate the significant speed-ups from the proposed algorithm, relative to the other distributed methods (Section 6.6). Finally, as a further implication of our results, we propose a novel distributed preconditioning method (Section 6.7), which can be used to improve the convergence rate of distributed gradient-based methods.

6.2 The Setup

We consider the problem of solving a large-scale system of linear equations

$$Ax = b, \tag{6.1}$$

where $A \in \mathbb{R}^{N\times n}$, $x \in \mathbb{R}^n$, and $b \in \mathbb{R}^N$. While we will generally take $N \geq n$, we will assume that the system has a unique solution. For this reason, we will most often consider the square case ($N = n$). The case where $N < n$, and there are multiple (infinitely many) solutions, is discussed in Section 6.5.

As mentioned before, for large-scale problems (when $N, n \gg 1$), it is highly desirable, or even necessary, to solve the problem in a distributed fashion. Assuming we have

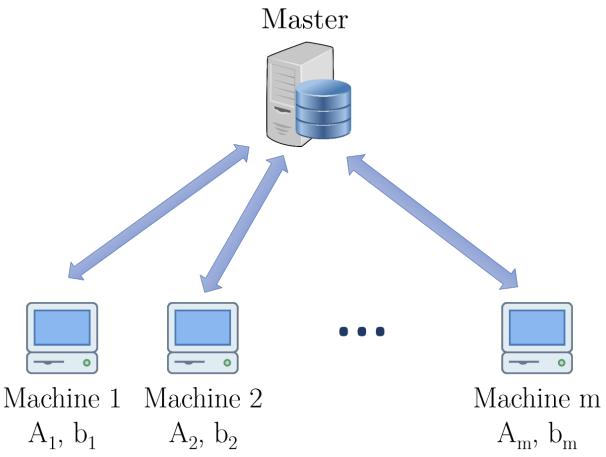


Figure 6.1: Schematic representation of the taskmaster and the m machines/cores. Each machine i has only a subset of the equations, i.e., $[A_i, b_i]$.

m machines (as in Figure 6.1), the equations can be partitioned so that each machine gets a disjoint subset of them. In other words, we can write (6.1) as

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

where each machine i receives $[A_i, b_i]$. In some applications, the data may already be stored on different machines in such a fashion. For the sake of simplicity, we assume that m divides N, and that the equations are distributed evenly among the machines, so that each machine gets $p = \frac{N}{m}$ equations. Therefore $A_i \in \mathbb{R}^{p \times n}$ and $b_i \in \mathbb{R}^p$ for every $i = 1, \dots m$. It is helpful to think of p as being relatively small compared to n. In fact, each machine has a system of equations which is highly under-determined.

6.3 Accelerated Projection-Based Consensus

6.3.1 The Algorithm

Each machine i can certainly find a solution (among infinitely many) to its own highly under-determined system of equations $A_i x = b_i$, with simply $O(p^3)$ computations. We denote this initial solution by $x_i(0)$. Clearly adding any vector in the right nullspace of A_i to $x_i(0)$ will yield another viable solution. The challenge is to find vectors in the nullspaces of each of the A_i's in such a way that all the solutions for

different machines coincide.

At each iteration t, the master provides the machines with an estimate of the solution, denoted by $\bar{x}(t)$. Each machine then updates its value $x_i(t)$ by projecting its difference from the estimate onto the nullspace, and taking a weighted step in that direction (which behaves as a "momentum"). Mathematically

$$x_i(t+1) = x_i(t) + \gamma P_i(\bar{x}(t) - x_i(t)),$$

where

$$P_i = I - A_i^T(A_i A_i^T)^{-1} A_i \tag{6.2}$$

is the projection matrix onto the nullspace of A_i (it is easy to check that $A_i P_i = 0$ and $P_i^2 = P_i$).

Although this might bear some resemblance to the block Cimmino method because of the projection matrices, APC has a much faster convergence rate than the block Cimmino method (i.e., convergence time smaller by a square root), as will be shown in Section 6.4. Moreover, it turns out that the block Cimmino method is in fact a special case of APC for $\gamma = 1$ (Section 6.4.5).

The update rule of $x_i(t+1)$ described above can be also thought of as the solution to an optimization problem with two terms: the distance from the global estimate $\bar{x}(t)$, and the distance from the previous solution $x_i(t)$. In other words, one can show that

$$\begin{aligned} x_i(t+1) = \underset{x_i}{\text{argmin}} \quad & \|x_i - \bar{x}(t)\|^2 + \frac{1-\gamma}{\gamma}\|x_i - x_i(t)\|^2 \\ \text{s.t.} \quad & A_i x_i = b_i \end{aligned}$$

The second term in the objective is what distinguishes this method from the block Cimmino method. If one sets γ equal to 1 (which is the reduction to the block Cimmino method), the second term disappears altogether, and the update no longer depends on $x_i(t)$. As we will show, this can have a dramatic impact on the convergence rate.

After each iteration, the master collects the updated values $x_i(t+1)$ to form a new estimate $\bar{x}(t+1)$. A plausible choice for this is to simply take the average of the values as the new estimate, i.e., $\bar{x}(t+1) = \frac{1}{m}\sum_{i=1}^m x_i(t+1)$. This update works, and is what appears both in ADMM and in the consensus method of [134, 146]. But it turns out that it is extremely slow. Instead, we take an affine combination of the average and the previous estimate as

$$\bar{x}(t+1) = \frac{\eta}{m}\sum_{i=1}^m x_i(t+1) + (1-\eta)\bar{x}(t),$$

Algorithm 4 APC: Accelerated Projection-Based Consensus (For solving $Ax = b$ distributedly)

Input: data $[A_i, b_i]$ on each machine $i = 1, \dots m$, parameters η, γ
Initialization: on each machine i, find a solution $x_i(0)$ (among infinitely many) to $A_i x = b_i$.
at the master, compute $\bar{x}(0) \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i(0)$
for $t = 1$ **to** T **do**
  for each machine i **parallel do**
    $x_i(t) \leftarrow x_i(t-1) + \gamma P_i(\bar{x}(t-1) - x_i(t-1))$
  end for
  at the master: $\bar{x}(t) \leftarrow \frac{\eta}{m} \sum_{i=1}^{m} x_i(t) + (1-\eta)\bar{x}(t-1)$
end for

which introduces a one-step memory, and again behaves as a momentum.

The resulting update rule is therefore

$$x_i(t+1) = x_i(t) + \gamma P_i(\bar{x}(t) - x_i(t)), \quad i \in [m], \tag{6.3a}$$

$$\bar{x}(t+1) = \frac{\eta}{m} \sum_{i=1}^{m} x_i(t+1) + (1-\eta)\bar{x}(t), \tag{6.3b}$$

which leads to Algorithm 4.

6.3.2 Convergence Analysis

We analyze the convergence of the proposed algorithm and prove that it has linear convergence (i.e., the error decays exponentially), with no additional assumption imposed. We also derive the rate of convergence explicitly.

Let us define the matrix $X \in \mathbb{R}^{n \times n}$ as

$$X \triangleq \frac{1}{m} \sum_{i=1}^{m} A_i^T (A_i A_i^T)^{-1} A_i. \tag{6.4}$$

As it will become clear soon, the condition number of this matrix predicts the behavior of the algorithm. Note that since the eigenvalues of the projection matrix P_i are all 0 and 1, for every i, the eigenvalues of X are all between 0 and 1. Denoting the eigenvalues of X by μ_i, $0 \leq \mu_{\min} \triangleq \mu_n \leq \cdots \leq \mu_1 \triangleq \mu_{\max} \leq 1$. Let us define complex quadratic polynomials $p_i(\lambda)$ characterized by γ and η as

$$p_i(\lambda; \gamma, \eta) \triangleq \lambda^2 + \left(-\eta\gamma(1-\mu_i) + \gamma - 1 + \eta - 1\right)\lambda + (\gamma - 1)(\eta - 1) \tag{6.5}$$

for $i = 1, \ldots, n$. Further, define set S as the collection of pairs $\gamma \in [0, 2]$ and $\eta \in \mathbb{R}$ for which the largest magnitude solution of $p_i(\lambda) = 0$ among every i is less than 1, i.e.,

$$S = \{(\gamma, \eta) \in [0, 2] \times \mathbb{R} \mid \text{roots of } p_i \text{ have magnitude less than 1 for all } i\}. \quad (6.6)$$

The following result summarizes the convergence behavior of the proposed algorithm.

Theorem 31. *Algorithm 4 converges to the true solution as fast as ρ^t converges to 0, as $t \to \infty$, for some $\rho \in (0, 1)$, if and only if $(\gamma, \eta) \in S$. Furthermore, the optimal rate of convergence is*

$$\rho = \frac{\sqrt{\kappa(X)} - 1}{\sqrt{\kappa(X)} + 1} \approx 1 - \frac{2}{\sqrt{\kappa(X)}}, \quad (6.7)$$

where $\kappa(X) = \frac{\mu_{\max}}{\mu_{\min}}$ is the condition number of X, and the optimal parameters (γ^*, η^*) are the solution to the following equations:

$$\begin{cases} \mu_{\max}\eta\gamma = (1 + \sqrt{(\gamma - 1)(\eta - 1)})^2, \\ \mu_{\min}\eta\gamma = (1 - \sqrt{(\gamma - 1)(\eta - 1)})^2. \end{cases}$$

Proof. Let x^* be the solution of $Ax = b$. To make the analysis easier, we define error vectors with respect to x^* as $e_i(t) = x_i(t) - x^*$ for all $i = 1 \ldots m$, and $\bar{e}(t) = \bar{x}(t) - x^*$, and work with these vectors. Using this notation, Eq. (6.3a) can be rewritten as

$$e_i(t + 1) = e_i(t) + \gamma P_i(\bar{e}(t) - e_i(t)), \quad i = 1, \ldots, m.$$

Note that both x^* and $x_i(t)$ are solutions to $A_i x = b_i$. Therefore, their difference, which is $e_i(t)$, is in the nullspace of A_i, and it remains unchanged under projection onto the nullspace. As a result, $P_i e_i(t) = e_i(t)$, and we have

$$e_i(t + 1) = (1 - \gamma)e_i(t) + \gamma P_i \bar{e}(t), \quad i = 1, \ldots, m. \quad (6.8)$$

Similarly, the recursion (6.3b) can be expressed as

$$\bar{e}(t + 1) = \frac{\eta}{m} \sum_{i=1}^{m} e_i(t + 1) + (1 - \eta)\bar{e}(t),$$

which using (6.8) becomes

$$\begin{aligned}\bar{e}(t+1) &= \frac{\eta}{m}\sum_{i=1}^{m}\left((1-\gamma)e_i(t)+\gamma P_i\bar{e}(t)\right)+(1-\eta)\bar{e}(t)\\ &= \frac{\eta(1-\gamma)}{m}\sum_{i=1}^{m}e_i(t)+\left(\frac{\eta\gamma}{m}\sum_{i=1}^{m}P_i+(1-\eta)I_n\right)\bar{e}(t). \qquad (6.9)\end{aligned}$$

It is relatively easy to check that in the steady state, the recursions (6.8), (6.9) become

$$\begin{cases}P_i\bar{e}(\infty)=e_i(\infty), \quad i=1,\dots,m\\ \bar{e}(\infty)=\frac{1}{m}\sum_{i=1}^{m}P_i\bar{e}(\infty),\end{cases}$$

which, because of $\frac{1}{m}\sum_{i=1}^{m}P_i = I-\frac{1}{m}\sum_{i=1}^{m}A_i^T(A_iA_i^T)^{-1}A_i = I-X$, implies $\bar{e}(\infty) = e_1(\infty)=\cdots=e_m(\infty)=0$, if $\mu_{\min}\neq 0$.

Now let us stack up all the m vectors e_i along with the average $\bar{e}$ together, as a vector $e(t)^T=[e_1(t)^T,e_2(t)^T,\dots,e_m(t)^T,\bar{e}(t)^T]\in\mathbb{R}^{(m+1)n}$. The update rule can be expressed as:

$$\begin{bmatrix}e_1(t+1)\\ \vdots\\ e_m(t+1)\\ \bar{e}(t+1)\end{bmatrix}=\begin{bmatrix}(1-\gamma)I_{mn} & \gamma\begin{bmatrix}P_1\\ \vdots\\ P_m\end{bmatrix}\\ \frac{\eta(1-\gamma)}{m}\begin{bmatrix}I_n\dots I_n\end{bmatrix} & M\end{bmatrix}\begin{bmatrix}e_1(t)\\ \vdots\\ e_m(t)\\ \bar{e}(t)\end{bmatrix}, \qquad (6.10)$$

where $M=\frac{\eta\gamma}{m}\sum_{i=1}^{m}P_i+(1-\eta)I_n$.

The convergence rate of the algorithm is determined by the spectral radius (largest magnitude eigenvalue) of the $(m+1)n\times(m+1)n$ block matrix in (6.10). The eigenvalues λ_i of this matrix are indeed the solutions to the following characteristic equation.

$$\det\begin{bmatrix}(1-\gamma-\lambda)I_{mn} & \gamma\begin{bmatrix}P_1\\ \vdots\\ P_m\end{bmatrix}\\ \frac{\eta(1-\gamma)}{m}\begin{bmatrix}I_n\dots I_n\end{bmatrix} & \frac{\eta\gamma}{m}\sum_{i=1}^{m}P_i+(1-\eta-\lambda)I_n\end{bmatrix}=0.$$

Using the Schur complement and properties of determinant, the characteristic

equation can be simplified as follows.

$$
\begin{aligned}
0 &= (1-\gamma-\lambda)^{mn}\times \\
&\quad \det\left(\frac{\eta\gamma}{m}\sum_{i=1}^{m}P_i + (1-\eta-\lambda)I_n - \frac{\eta(1-\gamma)\gamma}{(1-\gamma-\lambda)m}\sum_{i=1}^{m}P_i\right) \\
&= (1-\gamma-\lambda)^{mn}\times \\
&\qquad \det\left(\frac{\eta\gamma}{m}(1-\frac{1-\gamma}{1-\gamma-\lambda})\sum_{i=1}^{m}P_i + (1-\eta-\lambda)I_n\right) \\
&= (1-\gamma-\lambda)^{mn}\times \\
&\qquad \det\left(\frac{-\eta\gamma\lambda}{(1-\gamma-\lambda)m}\sum_{i=1}^{m}P_i + (1-\eta-\lambda)I_n\right) \\
&= (1-\gamma-\lambda)^{(m-1)n}\times \\
&\qquad \det\left(-\eta\gamma\lambda\frac{\sum_{i=1}^{m}P_i}{m} + (1-\gamma-\lambda)(1-\eta-\lambda)I_n\right).
\end{aligned}
$$

Therefore, there are $(m-1)n$ eigenvalues equal to $1-\gamma$, and the remaining $2n$ eigenvalues are the solutions to

$$
\begin{aligned}
0 &= \det\left(-\eta\gamma\lambda(I-X) + (1-\gamma-\lambda)(1-\eta-\lambda)I\right) \\
&= \det\left(\eta\gamma\lambda X + \left((1-\gamma-\lambda)(1-\eta-\lambda) - \eta\gamma\lambda\right)I\right).
\end{aligned}
$$

Whenever we have dropped the subscript of the identity matrix, it is of size n.

Recall that the eigenvalues of X are denoted by $\mu_i,\ i=1,\ldots,n$. Therefore, the eigenvalues of $\eta\gamma\lambda X + \left((1-\gamma-\lambda)(1-\eta-\lambda) - \eta\gamma\lambda\right)I$ are $\eta\gamma\lambda\mu_i + (1-\gamma-\lambda)(1-\eta-\lambda) - \eta\gamma\lambda,\ i=1,\ldots,n$. The above determinant can then be written as the product of the eigenvalues of the matrix inside it, as

$$
0 = \prod_{i=1}^{n}\eta\gamma\lambda\mu_i + (1-\gamma-\lambda)(1-\eta-\lambda) - \eta\gamma\lambda.
$$

Therefore, there are two eigenvalues $\lambda_{i,1}, \lambda_{i,2}$ as the solution to the quadratic equation

$$
\lambda^2 + (-\eta\gamma(1-\mu_i) + \gamma - 1 + \eta - 1)\lambda + (\gamma-1)(\eta-1) = 0
$$

for every $i=1,\ldots,n$, which will constitute the $2n$ eigenvalues. When all these eigenvalues, along with $1-\gamma$, are less than 1, the error converges to zero as ρ^t, with ρ being the largest magnitude eigenvalue (spectral radius). Therefore, Algorithm 4

converges to the true solution x^* as fast as ρ^t converges to 0, as $t \to \infty$, if and only if $(\gamma, \eta) \in S$.

The optimal rate of convergence is achieved when the spectral radius is minimum. For that to happen, all the above eigenvalues should be complex and have magnitude $|\lambda_{i,1}| = |\lambda_{i,2}| = \sqrt{(\gamma-1)(\eta-1)} = \rho$. It implies that we should have

$$(\gamma + \eta - \eta\gamma(1-\mu_i) - 2)^2 \leq 4(\gamma-1)(\eta-1), \quad \forall i,$$

or equivalently

$$-2\sqrt{(\gamma-1)(\eta-1)} \leq \gamma + \eta - \eta\gamma(1-\mu_i) \leq 2\sqrt{(\gamma-1)(\eta-1)}$$

for all i. The expression in the middle is an increasing function of μ_i, and therefore for the above bounds to hold, it is enough for the lower bound to hold for the $\mu_{\min}$ and the upper bound to hold for $\mu_{\max}$, i.e.,

$$\begin{cases} \gamma + \eta - \eta\gamma(1-\mu_{\max}) - 2 = 2\sqrt{(\gamma-1)(\eta-1)} \\ 2 + \eta\gamma(1-\mu_{\min}) - \gamma - \eta = 2\sqrt{(\gamma-1)(\eta-1)} \end{cases},$$

which can be massaged and expressed as

$$\begin{cases} \mu_{\max}\eta\gamma = (1 + \sqrt{(\gamma-1)(\eta-1)})^2 = (1+\rho)^2 \\ \mu_{\min}\eta\gamma = (1 - \sqrt{(\gamma-1)(\eta-1)})^2 = (1-\rho)^2 \end{cases}.$$

Dividing the above two equations implies $\kappa(X) = \frac{(1+\rho)^2}{(1-\rho)^2}$, which results in the optimal rate of convergence being

$$\rho = \frac{\sqrt{\kappa(X)} - 1}{\sqrt{\kappa(X)} + 1},$$

and that concludes the proof. □

We should remark that, while in theory, the optimal values of γ and η depend on the values of the smallest and largest eigenvalues of X, in practice, one will almost never compute these eigenvalues. Rather, one will use surrogate heuristics (such as using the eigenvalues of an appropriate-size random matrix) to choose the step size. (In fact, the other methods, such as distributed gradient descent and its variants, have the same issue as well.)

Table 6.1: A summary of the convergence rates of different methods. DGD: Distributed Gradient Descent, D-NAG: Distributed Nesterov's Accelerated Gradient Descent, D-HBM: Distributed Heavy-Ball Method, Mou et al: Consensus algorithm of [146], B-Cimmino: Block Cimmino Method, APC: Accelerated Projection-based Consensus. The smaller the convergence rate is, the faster is the method. Note that $\rho_{\text{GD}} \geq \rho_{\text{NAG}} \geq \rho_{\text{HBM}}$ and $\rho_{\text{Mou}} \geq \rho_{\text{Cim}} \geq \rho_{\text{APC}}$.

DGD	D-NAG	D-HBM	Mou et al.	B-Cimmino	APC (proposed)
$\frac{\kappa(A^TA)-1}{\kappa(A^TA)+1}$ $\approx 1-\frac{2}{\kappa(A^TA)}$	$1-\frac{2}{\sqrt{3\kappa(A^TA)+1}}$	$\frac{\sqrt{\kappa(A^TA)}-1}{\sqrt{\kappa(A^TA)}+1}$ $\approx 1-\frac{2}{\sqrt{\kappa(A^TA)}}$	$1-\mu_{\min}(X)$	$\frac{\kappa(X)-1}{\kappa(X)+1}$ $\approx 1-\frac{2}{\kappa(X)}$	$\frac{\sqrt{\kappa(X)}-1}{\sqrt{\kappa(X)}+1}$ $\approx 1-\frac{2}{\sqrt{\kappa(X)}}$

6.3.3 Computation and Communication Complexity

In addition to the convergence rate, or equivalently the number of iterations until convergence, one needs to consider the computational complexity per iteration. At each iteration, since $P_i = I_n - A_i^T(A_iA_i^T)^{-1}A_i$, and A_i is $p \times n$, each machine has to do the following two matrix-vector multiplications: (1) $A_i(x_i(t) - \bar{x}(t))$, which takes pn scalar multiplications, and (2) $\left(A_i^T(A_iA_i^T)^{-1}\right)$ times the vector from the previous step, which takes another np operations (the pseudoinverse $A_i^T(A_iA_i^T)^{-1}$ is computed only once). Thus the overall computational complexity of each iteration is $2pn$.

We should remark that the computation done at each machine during each iteration is essentially a projection, which has condition number one and is as numerically stable as a matrix vector multiplication can be.

Finally, the communication cost of the algorithm, per iteration, is as follows. After computing the update, each of the m machines sends an n-dimensional vector to the master, and receives back another n-dimensional vector, which is the new average.

As we will see, the per-iteration computation and communication complexity of the other algorithms are similar to APC; however, APC requires fewer iterations, because of its faster rate of convergence.

6.4 Comparison with Related Methods

6.4.1 Distributed Gradient Descent (DGD)

As mentioned earlier, (6.1) can also be viewed as an optimization problem of the form

$$\underset{x}{\text{minimize}} \ \|Ax - b\|^2,$$

and since the objective is separable in the data, i.e., $\|Ax - b\|^2 = \sum_{i=1}^m \|A_ix - b_i\|^2$, generic distributed optimization methods such as distributed gradient descent apply

well to the problem.

The regular or full gradient descent has the update rule $x(t+1) = x(t) - \alpha A^T(Ax(t) - b)$, where $\alpha > 0$ is the step size or learning rate. The distributed version of gradient descent is one in which each machine i has only a subset of the equations $[A_i, b_i]$, and computes its own part of the gradient, which is $A_i^T(A_i x(t) - b_i)$. The updates are then collectively done as:

$$x(t+1) = x(t) - \alpha \sum_{i=1}^{m} A_i^T(A_i x(t) - b_i). \tag{6.11}$$

One can show that this also has linear convergence, and the rate of convergence is

$$\rho_{\text{GD}} = \frac{\kappa(A^T A) - 1}{\kappa(A^T A) + 1} \approx 1 - \frac{2}{\kappa(A^T A)}. \tag{6.12}$$

We should mention that since each machine needs to compute $A_i^T(A_i x(t) - b_i)$ at each iteration t, the computational complexity per iteration is $2pn$, which is identical to that of APC.

6.4.2 Distributed Nesterov's Accelerated Gradient Descent (D-NAG)

A popular variant of gradient descent is Nesterov's accelerated gradient descent [151], which has a memory term, and works as follows:

$$y(t+1) = x(t) - \alpha \sum_{i=1}^{m} A_i^T(A_i x(t) - b_i), \tag{6.13a}$$

$$x(t+1) = (1+\beta)y(t+1) - \beta y(t). \tag{6.13b}$$

One can show [125] that the optimal convergence rate of this method is

$$\rho_{\text{NAG}} = 1 - \frac{2}{\sqrt{3\kappa(A^T A) + 1}}, \tag{6.14}$$

which is improved over the regular distributed gradient descent (one can check that $\frac{\kappa(A^T A)-1}{\kappa(A^T A)+1} \geq 1 - \frac{2}{\sqrt{3\kappa(A^T A)+1}}$).

6.4.3 Distributed Heavy-Ball Method (D-HBM)

The heavy-ball method [169], otherwise known as the gradient descent with momentum, is another accelerated variant of gradient descent as follows:

$$z(t+1) = \beta z(t) + \sum_{i=1}^{m} A_i^T(A_i x(t) - b_i), \tag{6.15a}$$

$$x(t+1) = x(t) - \alpha z(t+1). \tag{6.15b}$$

It can be shown [125] that the optimal rate of convergence of this method is

$$\rho_{\text{HBM}} = \frac{\sqrt{\kappa(A^TA)} - 1}{\sqrt{\kappa(A^TA)} + 1} \approx 1 - \frac{2}{\sqrt{\kappa(A^TA)}}, \tag{6.16}$$

which is further improved over DGD and D-NAG ($\frac{\kappa(A^TA)-1}{\kappa(A^TA)+1} \geq 1 - \frac{2}{\sqrt{3\kappa(A^TA)+1}} \geq \frac{\sqrt{\kappa(A^TA)}-1}{\sqrt{\kappa(A^TA)}+1}$). This is similar to, but not the same as, the rate of convergence of APC. The difference is that the condition number of $A^TA = \sum_{i=1}^m A_i^TA_i$ is replaced with the condition number of $X = \sum_{i=1}^m A_i^T \left(A_iA_i^T\right)^{-1} A_i$ in APC. Given its structure as the sum of projection matrices, one may speculate that X has a much better condition number than A^TA. Indeed, our experiments with random, as well as real, data sets suggest that this is the case and that the condition number of X is often significantly better (see Table 6.2).

6.4.4 Alternating Direction Method of Multipliers (ADMM)

Alternating Direction Method of Multipliers (more specifically, consensus ADMM [185, 44]), is another generic method for solving optimization problems with separable cost function $f(x) = \sum_{i=1}^m f_i(x)$ distributedly, by defining additional local variables. Each machine i holds local variables $x_i(t) \in \mathbb{R}^n$ and $y_i(t) \in \mathbb{R}^n$, and the master's value is $\bar{x}(t) \in \mathbb{R}^n$, for any time t. For $f_i(x) = \frac{1}{2}\|A_ix - b_i\|^2$, the update rule of ADMM simplifies to

$$x_i(t+1) = (A_i^TA_i + \xi I_n)^{-1}(A_i^Tb_i - y_i(t) + \xi\bar{x}(t)), \quad i \in [m] \tag{6.17a}$$

$$\bar{x}(t+1) = \frac{1}{m}\sum_{i=1}^m x_i(t+1) \tag{6.17b}$$

$$y_i(t+1) = y_i(t) + \xi(x_i(t+1) - \bar{x}(t+1)), \quad i \in [m] \tag{6.17c}$$

It turns out that this method is very slow (and often unstable) in its native form for the application in hand. One can check that when system (6.1) has a solution, all the y_i variables converge to zero in steady state. Therefore, setting y_i's to zero can speed up the convergence significantly. We use this modified version in Section 6.6 for comparison.

We should also note that the computational complexity of ADMM is $O(pn)$ per iteration (the inverse is computed using matrix inversion lemma), which is again the same as that of gradient-type methods and APC.

6.4.5 Block Cimmino Method

The Block Cimmino method [69, 191, 11], which is a parallel method specifically for solving linear systems of equations, is perhaps the closest algorithm in spirit to APC. It is, in a way, a distributed implementation of the so-called Kaczmarz method [111]. The convergence of the Cimmino method is slower by an order in comparison with APC (its convergence time is the square of that of APC), and it turns out that APC includes this method as a special case when $\gamma = 1$.

The block Cimmino method is the following:

$$r_i(t) = A_i^+(b_i - A_i\bar{x}(t)), \quad i \in [m] \tag{6.18a}$$

$$\bar{x}(t+1) = \bar{x}(t) + \nu \sum_{i=1}^{m} r_i(t), \tag{6.18b}$$

where $A_i^+ = A_i^T(A_iA_i^T)^{-1}$ is the pseudoinverse of A_i.

Proposition 32. *The APC method (Algorithm 4) includes the block Cimmino method as a special case for $\gamma = 1$.*

Proof. When $\gamma = 1$, Eq. (6.3a) becomes

$$\begin{aligned} x_i(t+1) &= x_i(t) - P_i\left(x_i(t) - \bar{x}(t)\right) \\ &= x_i(t) - \left(I - A_i^T(A_iA_i^T)^{-1}A_i\right)\left(x_i(t) - \bar{x}(t)\right) \\ &= \bar{x}(t) + A_i^T(A_iA_i^T)^{-1}A_i(x_i - \bar{x}(t)) \\ &= \bar{x}(t) + A_i^T(A_iA_i^T)^{-1}(b_i - A_i\bar{x}(t)) \end{aligned}$$

In the last equation, we used the fact that x_i is always a solution to $A_ix = b_i$. Notice that the above equation is no longer an "update" in the usual sense, i.e., $x_i(t+1)$ does not depend on $x_i(t)$ directly. This can be further simplified using the pseudoinverse of A_i, $A_i^+ = A_i^T(A_iA_i^T)^{-1}$ as

$$x_i(t+1) = \bar{x}(t) + A_i^+(b_i - A_i\bar{x}(t)).$$

It is then easy to see from the Cimmino's equation (6.18a) that

$$r_i(t) = x_i(t+1) - \bar{x}(t).$$

Therefore, the update (6.18b) can be expressed as

$$\begin{aligned}\bar{x}(t+1) &= \bar{x}(t) + \nu \sum_{i=1}^{m} r_i(t) \\ &= \bar{x}(t) + \nu \sum_{i=1}^{m} (x_i(t+1) - \bar{x}(t)) \\ &= (1 - m\nu)\bar{x}(t) + \nu \sum_{i=1}^{m} x_i(t+1),\end{aligned}$$

which is nothing but the same update rule as in (6.3b) with $\eta = m\nu$. □

It is not hard to show that optimal rate of convergence of the Cimmino method is

$$\rho_{\text{Cim}} = \frac{\kappa(X) - 1}{\kappa(X) + 1} \approx 1 - \frac{2}{\kappa(X)}, \tag{6.19}$$

which is slower (by an order) than that of APC ($\frac{\sqrt{\kappa(X)}-1}{\sqrt{\kappa(X)}+1} \approx 1 - \frac{2}{\sqrt{\kappa(X)}}$).

6.4.6 Consensus Algorithm of Mou et al.

As mentioned earlier, a projection-based consensus algorithm for solving linear systems over a network was recently proposed by Mou et al. [146, 134]. For the master-worker setting studied here, the corresponding network would be a clique, and the algorithm reduces to

$$x_i(t+1) = x_i(t) + P_i\left(\frac{1}{m}\left(\sum_{j=1}^{m} x_j(t)\right) - x_i(t)\right), \quad i \in [m], \tag{6.20}$$

which is transparently equivalent to APC with $\gamma = \eta = 1$:

$$\begin{aligned}x_i(t+1) &= x_i(t) + P_i(\bar{x}(t) - x_i(t)), \quad i \in [m], \\ \bar{x}(t+1) &= \frac{1}{m}\sum_{i=1}^{m} x_i(t+1),\end{aligned}$$

It is straightforward to show that the rate of convergence in this case is

$$\rho_{\text{Mou}} = 1 - \mu_{\min}(X) \tag{6.21}$$

which is much slower than the block Cimmino method and APC. One can easily check that

$$1 - \mu_{\min}(X) \geq \frac{\kappa(X) - 1}{\kappa(X) + 1} \geq \frac{\sqrt{\kappa(X)} - 1}{\sqrt{\kappa(X)} + 1}.$$

Even though this algorithm is slow, it is useful for applications where a fully-distributed (networked) solution is desired. Another networked algorithm for solving a least-squares problem has been recently proposed in [207].

A summary of the convergence rates of all the related methods discussed in this section is provided in Table 6.1.

6.5 Underdetermined System

In this section, we consider the case when $N < n$ and $\text{rank}(A) = N$, i.e., the system is underdetermined and there are infinitely many solutions. We prove that in this case, each machine still converges to "a" (global) solution, and further, all the machines converge to the same solution. The convergence is again linear (i.e., the error decays exponentially fast), and the rate of convergence is similar to the previous case.

Recall that the matrix $X \in \mathbb{R}^{n\times n}$ defined earlier can be written as

$$\begin{aligned} X &= \frac{1}{m}\sum_{i=1}^{m} A_i^T (A_i A_i^T)^{-1} A_i \\ &= \frac{1}{m} A^T \begin{bmatrix} (A_1 A_1^T)^{-1} & & \\ & \ddots & \\ & & (A_m A_m^T)^{-1} \end{bmatrix} A \end{aligned}$$

which is singular in this case.

We define a new matrix $Y \in \mathbb{R}^{N\times N}$

$$Y \triangleq \frac{1}{m} A A^T \begin{bmatrix} (A_1 A_1^T)^{-1} & & \\ & \ddots & \\ & & (A_m A_m^T)^{-1} \end{bmatrix} \tag{6.22}$$

which has the same nonzero eigenvalues as X.

Theorem 33. *Suppose $N < n$ and $\text{rank}(A) = N$. Each one of $x_1(t), \ldots, x_m(t), \bar{x}(t)$ in Algorithm 4 converges to a solution as fast as ρ^t converges to 0, as $t \to \infty$, for some $\rho \in (0, 1)$, if and only if $(\gamma, \eta) \in S$. Furthermore, the solutions converged to are the same. The optimal rate of convergence is*

$$\rho = \frac{\sqrt{\kappa(Y)} - 1}{\sqrt{\kappa(Y)} + 1} \approx 1 - \frac{2}{\sqrt{\kappa(Y)}}, \tag{6.23}$$

Table 6.2: A comparison between the condition numbers of A^TA and X for some examples. m is the number of machines/partitions. The condition number of X is typically much smaller (better). Remarkably, the difference is even more pronounced when A has non-zero mean.

	$\kappa(A^TA)$	$\kappa(X)$
100×100 $\mathcal{N}(0,1)$ (Gaussian)	7×10^4	2×10^4 $(m = 2)$ 4×10^4 $(m = 5)$ 5×10^4 $(m = 10)$ 6×10^4 $(m = 20)$
100×100 $\mathcal{N}(10,1)$ (Non-zero mean)	2×10^8	4×10^6 $(m = 2)$ 1×10^7 $(m = 5)$ 2×10^7 $(m = 10)$ 4×10^7 $(m = 20)$
100×100 $\mathcal{N}(0,10^2)$ (High variance)	8×10^5	2×10^5 $(m = 2)$ 5×10^5 $(m = 5)$ 6×10^5 $(m = 10)$ 7×10^5 $(m = 20)$
200×100 $\mathcal{N}(0,1)$ (Tall)	3×10^1	1×10^0 $(m = 2)$ 1×10^1 $(m = 5)$ 2×10^1 $(m = 10)$ 2×10^1 $(m = 20)$
100×100 exp(10) (Exponential)	9×10^5	1×10^4 $(m = 2)$ 4×10^4 $(m = 5)$ 1×10^5 $(m = 10)$ 2×10^5 $(m = 20)$
100×100 stable$(0.5, 0.5, 1, 0)$ (Heavy tail)	3×10^{15}	1×10^7 $(m = 2)$ 3×10^7 $(m = 5)$ 3×10^7 $(m = 10)$ 3×10^7 $(m = 20)$
Real example: QC324 (324×324)	2×10^7	1×10^5 $(m = 2)$ 3×10^5 $(m = 4)$ 6×10^5 $(m = 9)$ 6×10^5 $(m = 18)$ 8×10^5 $(m = 81)$
Real example: ORSIRR 1 (1030×1030)	6×10^9	4×10^7 $(m = 2)$ 5×10^7 $(m = 5)$ 5×10^7 $(m = 10)$ 6×10^7 $(m = 103)$ 6×10^7 $(m = 206)$
Real example: ASH608 (608×188)	11×10^0	8×10^0 $(m = 8)$ 10×10^0 $(m = 16)$ 10×10^0 $(m = 32)$ 11×10^0 $(m = 76)$

Table 6.3: A comparison between the optimal convergence time T ($= \frac{1}{-\log \rho}$) of different methods on real and synthetic examples. Boldface values show the smallest convergence time. QC324: Model of H_2^+ in an Electromagnetic Field. ORSIRR 1: Oil Reservoir Simulation. ASH608: Original Harwell sparse matrix test collection.

	DGD	D-NAG	D-HBM	M-ADMM	B-Cimmino	APC
QC324 (324×324)	1.22×10^7	4.28×10^3	2.47×10^3	1.07×10^7	3.10×10^5	$\mathbf{3.93 \times 10^2}$
ORSIRR 1 (1030×1030)	2.98×10^9	6.68×10^4	3.86×10^4	2.08×10^8	2.69×10^7	$\mathbf{3.67 \times 10^3}$
ASH608 (608×188)	5.67×10^0	2.43×10^0	1.64×10^0	1.28×10^1	4.98×10^0	$\mathbf{1.53 \times 10^0}$
Standard Gaussian (500×500)	1.76×10^7	5.14×10^3	2.97×10^3	1.20×10^6	1.46×10^7	$\mathbf{2.70 \times 10^3}$
Nonzero-Mean Gaussian (500×500)	2.22×10^{10}	1.82×10^5	1.05×10^5	8.62×10^8	9.29×10^8	$\mathbf{2.16 \times 10^4}$
Standard Tall Gaussian (1000×500)	1.58×10^1	4.37×10^0	2.78×10^0	4.49×10^1	1.13×10^1	$\mathbf{2.34 \times 10^0}$
Standard Fat Gaussian (400×500)	1.37×10^2	1.38×10^1	8.26×10^0	3.17×10^2	1.14×10^2	$\mathbf{7.54 \times 10^0}$

where $\kappa(Y) = \frac{\mu_{\max}}{\mu_{\min}}$ is the condition number of Y, and the optimal parameters (γ^*, η^*) are the solution to the following equations

$$\begin{cases} \mu_{\max}\eta\gamma = (1 + \sqrt{(\gamma-1)(\eta-1)})^2, \\ \mu_{\min}\eta\gamma = (1 - \sqrt{(\gamma-1)(\eta-1)})^2. \end{cases}$$

Proof. Let x^* be a solution to $Ax = b$. We define error vectors $e_i(t) = x_i(t) - x^*$ for all $i = 1 \dots m$, and $\bar{e}(t) = \bar{x}(t) - x^*$, as before, but this time show that $Ae_i(t) \to 0$ and $A\bar{e}(t) \to 0$. Recursion (6.3a) can be rewritten as

$$e_i(t+1) = e_i(t) + \gamma P_i(\bar{e}(t) - e_i(t)), \quad i = 1, \dots, m,$$

as before. Since both x^* and $x_i(t)$ are solutions to $A_i x = b_i$, their difference $e_i(t)$ is in the nullspace of A_i, and it remains unchanged under projection onto the nullspace. As a result, $P_i e_i(t) = e_i(t)$, and we have

$$e_i(t+1) = (1-\gamma)e_i(t) + \gamma P_i \bar{e}(t), \quad i = 1, \dots, m. \tag{6.24}$$

Similarly, the recursion (6.3b) can be expressed as

$$\begin{aligned}\bar{e}(t+1) &= \frac{\eta}{m}\sum_{i=1}^{m} e_i(t+1) + (1-\eta)\bar{e}(t) \\ &= \frac{\eta}{m}\sum_{i=1}^{m} ((1-\gamma)e_i(t) + \gamma P_i\bar{e}(t)) + (1-\eta)\bar{e}(t) \\ &= \frac{\eta(1-\gamma)}{m}\sum_{i=1}^{m} e_i(t) + \left(\frac{\eta\gamma}{m}\sum_{i=1}^{m} P_i + (1-\eta)I_n\right)\bar{e}(t),\end{aligned}$$

as before.

Multiplying the recursions by A, we have

$$Ae_i(t+1) = (1-\gamma)Ae_i(t) + \gamma AP_i\bar{e}(t), \quad i = 1,\dots,m,$$

and

$$A\bar{e}(t+1) = \frac{\eta(1-\gamma)}{m}\sum_{i=1}^{m} Ae_i(t) + \left(\frac{\eta\gamma}{m}\sum_{i=1}^{m} AP_i + (1-\eta)A\right)\bar{e}(t)$$

Note that $P_i = I_n - A_i^T(A_iA_i^T)^{-1}A_i$, and we can express A_i as $A_i = \begin{bmatrix}0_{p\times p} & \dots & I_p & \dots & 0_{p\times p}\end{bmatrix} A = E_iA$, where E_i is a $p \times N$ matrix with an identity at its i-th block and zero everywhere else. Therefore, we have $AP_i = A - AA_i^T(A_iA_i^T)^{-1}E_iA = (I_N - AA_i^T(A_iA_i^T)^{-1}E_i)A$, and the recursions become

$$Ae_i(t+1) = (1-\gamma)Ae_i(t) + \gamma(I_N - AA_i^T(A_iA_i^T)^{-1}E_i)A\bar{e}(t),$$

for $i = 1,\dots,m$, and

$$\begin{aligned}A\bar{e}(t+1) = \frac{\eta(1-\gamma)}{m}\sum_{i=1}^{m} Ae_i(t) \\ + \left(\frac{\eta\gamma}{m}\sum_{i=1}^{m}(I_N - AA_i^T(A_iA_i^T)^{-1}E_i) + (1-\eta)I_N\right)A\bar{e}(t).\end{aligned}$$

Stacking up all the m vectors Ae_i along with $A\bar{e}$, as an $(m+1)N$-dimensional vector, results in

$$\begin{bmatrix}Ae_1(t+1)\\ \vdots \\ Ae_m(t+1)\\ A\bar{e}(t+1)\end{bmatrix} = \begin{bmatrix}(1-\gamma)I_{mN} & \gamma\begin{bmatrix}P_1'\\ \vdots\\ P_m'\end{bmatrix}\\ \frac{\eta(1-\gamma)}{m}\begin{bmatrix}I_N \dots I_N\end{bmatrix} & M'\end{bmatrix}\begin{bmatrix}Ae_1(t)\\ \vdots \\ Ae_m(t)\\ A\bar{e}(t)\end{bmatrix},$$

where $M' = \frac{\eta\gamma}{m}\sum_{i=1}^{m} P_i' + (1-\eta)I_N$ and $P_i' = I_N - AA_i^T(A_iA_i^T)^{-1}E_i$.

The convergence rate of the algorithm is determined by the spectral radius (largest magnitude eigenvalue) of this $(m+1)N \times (m+1)N$ matrix. The eigenvalues λ_i of this matrix are the solutions to the following characteristic equation.

$$\det \begin{bmatrix} (1-\gamma-\lambda)I_{mN} & \gamma \begin{bmatrix} P_1' \\ \vdots \\ P_m' \end{bmatrix} \\ \frac{\eta(1-\gamma)}{m}\begin{bmatrix} I_N \dots I_N \end{bmatrix} & \frac{\eta\gamma}{m}\sum_{i=1}^{m} P_i' + (1-\eta-\lambda)I_N \end{bmatrix} = 0.$$

Similar as in the proof of Theorem 31, using the Schur complement and properties of determinant, the characteristic equation can be simplified as follows:

$$\begin{aligned} 0 &= (1-\gamma-\lambda)^{mN} \det\left(\frac{\eta\gamma}{m}\sum_{i=1}^{m} P_i' + (1-\eta-\lambda)I_N - \frac{\eta(1-\gamma)\gamma}{(1-\gamma-\lambda)m}\sum_{i=1}^{m} P_i'\right) \\ &= (1-\gamma-\lambda)^{mN} \det\left(\frac{\eta\gamma}{m}(1-\frac{1-\gamma}{1-\gamma-\lambda})\sum_{i=1}^{m} P_i' + (1-\eta-\lambda)I_N\right) \\ &= (1-\gamma-\lambda)^{mN} \det\left(\frac{-\eta\gamma\lambda}{(1-\gamma-\lambda)m}\sum_{i=1}^{m} P_i' + (1-\eta-\lambda)I_N\right) \\ &= (1-\gamma-\lambda)^{(m-1)N} \det\left(-\eta\gamma\lambda\frac{\sum_{i=1}^{m} P_i'}{m} + (1-\gamma-\lambda)(1-\eta-\lambda)I_N\right). \end{aligned}$$

Note that $\frac{1}{m}\sum_{i=1}^{m} P_i' = I_N - \frac{1}{m}\sum_{i=1}^{m} AA_i^T(A_iA_i^T)^{-1}E_i = I_N - [AA_1^T(A_1A_1^T)^{-1}, \dots, AA_m^T(A_mA_m^T)^{-1}] = I_N - Y$. There are $(m-1)N$ eigenvalues equal to $1-\gamma$, and the remaining $2N$ eigenvalues are the solutions to

$$\begin{aligned} 0 &= \det\left(-\eta\gamma\lambda(I-Y) + (1-\gamma-\lambda)(1-\eta-\lambda)I\right) \\ &= \det\left(\eta\gamma\lambda Y + \left((1-\gamma-\lambda)(1-\eta-\lambda) - \eta\gamma\lambda\right)I\right). \end{aligned}$$

Notice that this is exactly the same as the one in the proof of Theorem 31, with X replaced with Y.

It follows that the $Ae_1(t), \dots, Ae_m(t), A\bar{e}(t)$ converge to zero as fast as ρ^t if and only if $(\gamma, \eta) \in S$, and the optimal rate of convergence is

$$\rho = \frac{\sqrt{\kappa(Y)} - 1}{\sqrt{\kappa(Y)} + 1}.$$

Convergence of $Ae_1(t), \dots, Ae_m(t), A\bar{e}(t)$ to zero means that each machine and the master converge to a solution, but the solutions reached may not be the same.

What remains to show is that the only steady state is the "consensus steady state." From (6.3), it is easy to see that the steady state $x_1(\infty), \ldots, x_m(\infty), \bar{x}(\infty)$ satisfies the following equation.

$$\begin{cases} P_i(\bar{x}(\infty) - x_i(\infty)) = 0, \ i \in [m] \\ \bar{x}(\infty) = \frac{1}{m}\sum_{i=1}^{m} x_i(\infty) \end{cases} \tag{6.25}$$

which can be written in a matrix form as

$$\begin{bmatrix} P_1 & & & -P_1 \\ & \ddots & & \vdots \\ & & P_m & -P_m \\ -\frac{I_n}{m} & \ldots & -\frac{I_n}{m} & I_n \end{bmatrix} \begin{bmatrix} x_1(\infty) \\ \vdots \\ x_m(\infty) \\ \bar{x}(\infty) \end{bmatrix} = 0. \tag{6.26}$$

Notice that for any $v \in \mathbb{R}^n$, the vector $\begin{bmatrix} v^T & \ldots & v^T & v^T \end{bmatrix}^T$ is a solution to this equation, which corresponds to a consensus steady state $x_1(\infty) = \cdots = x_m(\infty) = \bar{x}(\infty) = v$. Therefore, the nullspace of the above matrix is at least n dimensional, or in other words, it has n zero eigenvalues. We will argue that this matrix has only n zero eigenvalues, and therefore any steady-state solution must be a consensus. To find the eigenvalues λ_i we have to solve the following characteristic equation.

$$\det \begin{bmatrix} P_1 - \lambda I & & & -P_1 \\ & \ddots & & \vdots \\ & & P_m - \lambda I & -P_m \\ -\frac{I}{m} & \ldots & -\frac{I}{m} & (1-\lambda)I \end{bmatrix} = 0.$$

Once again, using the Schur complement, we have

$$0 = \left(\prod_{i=1}^{m} \det(P_i - \lambda I)\right) \det\left((1-\lambda)I - \frac{1}{m}\sum_{i=1}^{m}(P_i - \lambda I)^{-1}P_i\right)$$

Note that $(P_i - \lambda I)^{-1}P_i = \frac{1}{1-\lambda}P_i$. Therefore, we can write

$$\begin{aligned} 0 &= \left(\prod_{i=1}^{m} \det(P_i - \lambda I)\right) \det\left((1-\lambda)I - \frac{1}{m}\sum_{i=1}^{m}\frac{1}{1-\lambda}P_i\right) \\ &= \left(\prod_{i=1}^{m} ((-\lambda)^p(1-\lambda)^{n-p})\right) \det\left((1-\lambda)I - \frac{1}{m}\sum_{i=1}^{m}\frac{1}{1-\lambda}P_i\right) \\ &= (-\lambda)^N(1-\lambda)^{mn-N} \det\left((1-\lambda)I - \frac{1}{m}\sum_{i=1}^{m}\frac{1}{1-\lambda}P_i\right) \end{aligned}$$

because P_i has p zero eigenvalues and $n - p$ one eigenvalues. Using the properties of determinant, we further have

$$\begin{aligned}
0 &= (-\lambda)^N (1-\lambda)^{(m-1)n-N} \det\left((1-\lambda)^2 I - \frac{1}{m}\sum_{i=1}^{m} P_i\right) \\
&= (-\lambda)^N (1-\lambda)^{(m-1)n-N} \det\Big((1-\lambda)^2 I - (I - X)\Big) \\
&= (-\lambda)^N (1-\lambda)^{(m-1)n-N} \det\Big((\lambda^2 - 2\lambda) I + X\Big) \\
&= (-\lambda)^N (1-\lambda)^{(m-1)n-N} \prod_{i=1}^{m} \Big(\lambda^2 - 2\lambda + \mu_i\Big)
\end{aligned}$$

Therefore, the $(m+1)n$ eigenvalues are as follows: N zero eigenvalues, $(m-1)n - N$ eigenvalues at 1, and the remaining $2n$ are $1 \pm \sqrt{1-\mu_i}$. Note that in the underdetermined case, X has $n - N$ zero eigenvalues. Therefore, $n - N$ of $1 \pm \sqrt{1-\mu_i}$ are zero. As a result, the overall number of zero eigenvalues is $N + (n - N) = n$. This implies that the nullity of the matrix in (6.26) is n and any steady-state solution must be a consensus solution, which completes the proof.

□

6.6 Experimental Results

In this section, we evaluate the proposed method (APC) by comparing it with the other distributed methods discussed throughout the chapter, namely DGD, D-NAG, D-HBM, modified ADMM, and block Cimmino methods. We use randomly-generated problems as well as real-world ones from the National Institute of Standards and Technology (NIST) repository, *Matrix Market* [141].

We first compare the rate of convergence of the algorithms, ρ, which is the spectral radius of the iteration matrix. To distinguish the differences, it is easier to compare the convergence time, which is defined as $T = \frac{1}{-\log \rho}$ ($\approx \frac{1}{1-\rho}$). We tune the parameters in all of the methods to their optimal values, to make the comparison between the methods fair. Also as mentioned before, all the algorithms have the same per-iteration computation and communication complexity. Table 6.3 shows the values of the convergence times for a number of synthetic and real-world problems with different sizes. It can be seen that APC has a much faster convergence, often by orders of magnitude. As expected from the analysis, the APC's closest competitor is the distributed heavy-ball method. Notably, in randomly-generated problems, when the mean is not zero, the gap is much larger.

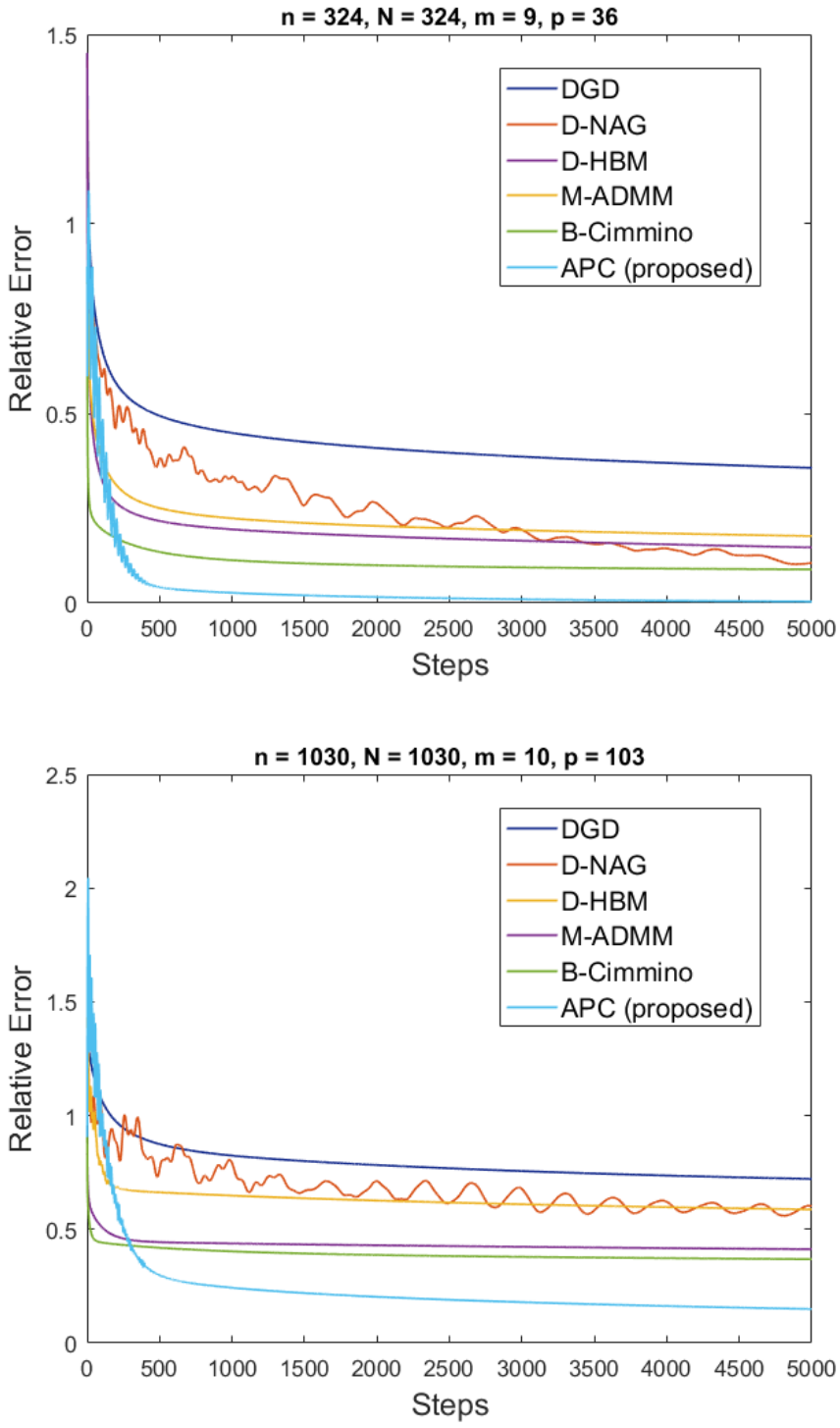


Figure 6.2: The decay of the error for different distributed algorithms, on two real problems from Matrix Market [141] (QC324: Model of H_2^+ in an Electromagnetic Field, and ORSIRR 1: Oil reservoir simulation). n = # of variables, N = # of equations, m = # of workers, p = # of equations per worker.

To further verify the performance of the proposed algorithm, we also run all the algorithms on multiple problems, and observe the actual decay of the error. Fig. 6.2 shows the relative error (the distance from the true solution, divided by the true solution, in ℓ_2 norm) for all the methods, on two examples from the repository. Again, to make the comparison fair, all the methods have been tuned to their optimal parameters. As one can see, APC outperforms the other methods by a wide margin, which is consistent with the order-of-magnitude differences in the convergence times of Table 6.3. We should also remark that initialization does not seem to affect the convergence behavior of our algorithm. Lastly, we should mention that our experiments on cases where there are missing updates ("straggler" machines) indicate that APC is at least as robust as the other algorithms to these effects, and the convergence curves look qualitatively the same as in Fig. 6.2.

6.7 A Distributed Preconditioning to Improve Gradient-Based Methods

The noticeable similarity between the optimal convergence rate of APC ($\frac{\sqrt{\kappa(X)}-1}{\sqrt{\kappa(X)}+1}$) and that of D-HBM ($\frac{\sqrt{\kappa(A^TA)}-1}{\sqrt{\kappa(A^TA)}+1}$) suggests that there might be a connection between the two. It turns out that there is, and we propose a *distributed preconditioning* for D-HBM, which makes it achieve the same convergence rate as APC. The algorithm works as follows.

Prior to starting the iterative process, each machine i can premultiply its own set of equations $A_ix = b_i$ by $(A_iA_i^T)^{-1/2}$, which can be done in parallel (locally) with $O(p^2n)$ operations. This transforms the global system of equations $Ax = b$ to a new one $Cx = d$, where

$$C = \begin{bmatrix} (A_1A_1^T)^{-1/2}A_1 \\ \vdots \\ (A_mA_m^T)^{-1/2}A_m \end{bmatrix},$$

and

$$d = \begin{bmatrix} (A_1A_1^T)^{-1/2}b_1 \\ \vdots \\ (A_mA_m^T)^{-1/2}b_m \end{bmatrix}.$$

The new system can then be solved using distributed heavy-ball method, which will achieve the same rate of convergence as APC, i.e., $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ where $\kappa = \kappa(C^TC) = \kappa(X)$.

6.8 Conclusion

We considered the problem of solving a large-scale system of linear equations by a taskmaster with the help of a number of computing machines/cores, in a distributed way. We proposed an accelerated projection-based consensus algorithm for this problem, and fully analyzed its convergence rate. Analytical and experimental comparisons with the other known distributed methods confirm significantly faster convergence of the proposed scheme. Finally, our analysis suggests a novel distributed preconditioning for improving the convergence of the distributed heavy-ball method to achieve the same theoretical performance as the proposed consensus-based method.

We should finally remark that while the setting studied here was a master-workers one, the same algorithm can be implemented in a networked setting where there is no central collector/master, using a "distributed averaging" approach ([200, 215]).

Chapter 7

CODED COMPUTATION FOR DISTRIBUTED GRADIENT DESCENT

[1] Wael Halbawi et al. "Improving Distributed Gradient Descent Using Reed-Solomon Codes". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. 2018, pp. 2027–2031. DOI: 10.1109/ISIT.2018.8437467.

Today's massively-sized datasets have made it necessary to often perform computations on them in a distributed manner. In principle, a computational task is divided into subtasks which are distributed over a cluster operated by a taskmaster. One issue faced in practice is the delay incurred due to the presence of slow machines, known as *stragglers*. Several schemes, including those based on replication, have been proposed in the literature to mitigate the effects of stragglers and more recently, those inspired by coding theory have begun to gain traction. In this chapter, we consider a distributed gradient descent setting suitable for a wide class of machine learning problems. We adopt the framework of Tandon et al. [197] and present a deterministic scheme that, for a prescribed per-machine computational effort, recovers the gradient from the least number of machines f theoretically permissible, via an $O(f^2)$ decoding algorithm. We also provide a theoretical delay model which can be used to minimize the expected waiting time per computation by optimally choosing the parameters of the scheme. Finally, we supplement our theoretical findings with numerical results that demonstrate the efficacy of the method and its advantages over competing schemes.

7.1 Introduction

With the size of today's datasets, due to high computation and/or memory requirements, it is virtually impossible to run large-scale learning tasks on a single machine; and even if that is possible, the learning process can be extremely slow due to its sequential nature. Therefore, it is highly desirable or, even necessary, to run the tasks in a distributed fashion on multiple machines/cores. For this reason, parallel and distributed computing has attracted a lot of attention in recent years from the machine learning, and other, communities [44, 173, 22, 228, 77].

When a task is divided among a number of machines, the "computation time" is

clearly reduced significantly, since the task is being processed in parallel rather than sequentially. However, the taskmaster has to wait for all the machines in order to be able to recover the exact desired computation. Therefore, in the face of substantial or heterogeneous delays, distributed computing may suffer from being slow, which defeats the purpose of the exercise. Several approaches have been proposed to tackle this problem. One naive yet common way, especially when the task consists of many iterations, is to not wait for all machines, and ignore the *straggling machines*. One may hope that in this way on average the taskmaster receives enough information from everyone; however, it is clear that the performance of the learning algorithm may be significantly impacted in many cases because of lost updates. An alternative and more appropriate way to resolve this issue, is to introduce some *redundancy* in the computation of the machines, in order to efficiently trade off computation time for less wait time, and to be able to recover the correct update using only a few machines. But the great challenge here is to design a clever scheme for distributing the task among the machines, such that the computation can be recovered using a few machines, independent of which machines they are.

Over the past few decades, coding theory has been developed to address similar challenges in other domains, and has had enormous success in many applications such as mobile communication, storage, data transmission, and broadcast systems. Despite the existence of a great set of tools developed in coding theory which can be used in many machine learning problems, researchers had not looked at this area until very recently [123, 197, 70, 128]. This work is aimed at bridging the gap between distributed machine learning and coding theory, by introducing a carefully designed coding scheme for efficiently distributing a learning task among a number of machines.

More specifically, we consider gradient-based methods for additively separable cost functions, which are the most common way of training any model in machine learning, and use coding to cleverly distribute each gradient iteration across n machines in an efficient way. To that end, we propose a deterministic construction based on Reed-Solomon codes [174] accompanied with an efficient decoder, which is used to recover the full gradient update from a fixed number of returning machines. Furthermore, we provide a new delay model based on heavy-tail distributions that also incorporates the time required for decoding. We analyze this model theoretically and use it to optimally pick our scheme's parameters. We compare the performance of our method on the MNIST dataset [121] with other approaches, namely: 1)

Ignoring the straggling machines [164], 2) Waiting for all the machines, and 3) GradientCoding as proposed by Tandon et al. [197]. Our numerical results show that, for the same training time, our scheme achieves better test errors.

7.1.1 Related Work

As mentioned earlier, coding theory in machine learning is a relatively new area. We summarize the recent related work here. Lee et al. [123] recently employed a coding-theoretic method in two specific distributed tasks, namely matrix multiplication and data shuffling. They showed significant speed-ups are possible in those two tasks by using coding. Dutta et al. [70] proposed a method that speeds up distributed matrix multiplication by sparsifying the inner products computed at each machine. Polynomial codes has been proposed by Yu et al. [220], which uses a carefully-designed Reed-Solomon code for matrix multiplication. They have shown that their framework achieves the minimum recovery threshold while allowing efficient decoding using polynomial interpolation. A coded MapReduce framework was introduced by Li et al in [128] which is used to facilitate data shuffling in distributed computing. The closest work to our framework is the work of Tandon et al. [197], which aims at mitigating the effect of stragglers in distributed gradient descent using Maximum-Distance Separable (MDS) codes. However, no analysis of computation time was provided. Furthermore, in their framework, along with the above-mentioned works, the decoding was assumed to be performed offline which might be impractical in certain settings. The recent work [172] includes a coding scheme that is similar to the one presented here. In particular, the authors use cyclic MDS codes to recover the exact full gradient in the presence of stragglers. In addition, the authors present a clever scheme, based on adjacency matrices of expander graphs, to compute an approximation of the gradient in the presence of stragglers. We mention that our scheme differs in the the way the workload is distributed across the different machines: we advocate a load-balanced approach in which every machine performs the same amount of work. We show that this is possible for any number of machines n and prespecified workload w.

7.1.2 Statement of Contributions

In this work, we make the following three main contributions.

1. We construct a deterministic coding scheme for efficiently distributing gradient descent over a given number of machines. Our scheme is optimal in the sense that it can recover the gradient from the smallest possible number of returning

machines, f, given a prespecified computational effort per machine.

2. We provide an efficient online decoder, with time complexity $O(f^2)$ for recovering the gradient from any f machines, which is faster than the best known method [197], $O(f^3)$.

3. We analyze the total computation time, and provide a method for finding the optimal coding parameters. We consider heavy-tailed delays, which have been widely observed in CPU job runtimes in practice [124, 93, 94].

The rest of the chapter is organized as follows. In Section 7.2, we describe the problem setup and explain the design objectives in detail. Section 7.3, provides the construction of our coding scheme, using the idea of balanced Reed-Solomon codes. Our efficient online decoder is presented in Section 7.4. We then characterize the total computation time, and describe the optimal choice of coding parameters, in Section 7.5. Finally, we provide our numerical results in Section 7.6, and conclude in Section 7.7.

7.2 Preliminaries

7.2.1 Problem Setup

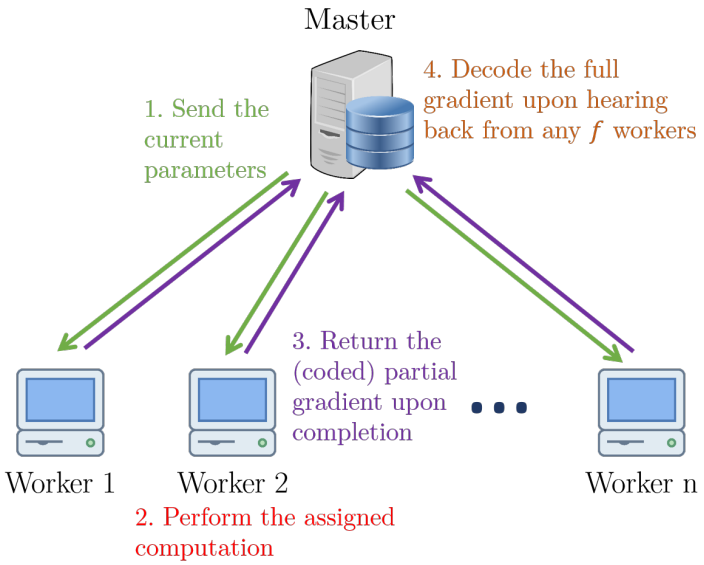


Figure 7.1: Schematic representation of the taskmaster and the n workers.

Consider a setting where there is a taskmaster M and there are n workers (computing machines) $W_1, W_2, \ldots, W_n$ interacting with the taskmaster, as in Fig. 7.1. The

master intends to train a model using gradient descent by distributing the gradient updates amongst the workers. More precisely, consider a typical scenario, where we want to learn parameters $\beta \in \mathbb{R}^p$ by minimizing a generic loss function $L(\mathcal{D}; \beta)$ over a given dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. The loss function can be expressed as the sum of the losses for individual data points, i.e. $L(\mathcal{D}; \beta) = \sum_{i=1}^N \ell(x_i, y_i; \beta)$. Therefore, the full gradient, with respect to β, is given by

$$\nabla L(\mathcal{D}; \beta) = \sum_{i=1}^{N} \nabla \ell(x_i, y_i; \beta). \tag{7.1}$$

The data can be divided into k (disjoint) chunks $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ of size $\frac{N}{k}$, and clearly the gradient can also be written as $\nabla L(\mathcal{D}; \beta) = \sum_{i=1}^k \sum_{(x,y)\in\mathcal{D}_i} \nabla \ell(x, y; \beta)$. Define $g_i := \sum_{(x,y)\in\mathcal{D}_i} \nabla \ell(x, y; \beta)$ as the partial gradient of chunk i for every i, and $g^T := [g_1^T, g_2^T, \ldots, g_k^T]$, where g_i is a row vector of length p. Therefore $\nabla L(\mathcal{D}; \beta) = \mathbf{1}_{1\times k} g$. Now suppose each worker W_i is assigned w data partitions $\{\mathcal{D}_{i_1}, \ldots, \mathcal{D}_{i_w}\}$, on which it computes the partial gradients $\{g_{i_1}, g_{i_2}, \ldots, g_{i_w}\}$. Note that the "redundancy" in computation is introduced here, since each chunk is allowed to be assigned to multiple workers. Each worker then has to compute its partial gradients, and return a prespecified *linear combination* of them to the master.

As it will be explained in detail, k and w are to be chosen in such a way that the total computation time is minimized. For a fixed k and w, we want to be able to recover the gradient using the linear combinations received from the fastest f machines at the master (or equivalently tolerate $s := n - f$ stragglers). Note that we do not assume any prior knowledge about the stragglers, i.e., we shall design a scheme that enables master to recover the gradient from any set of f machines. It is known [197] that for any fixed k and w, an upper-bound on the number of stragglers that *any scheme* can tolerate is:

$$s \leq \left\lfloor \frac{wn}{k} \right\rfloor - 1. \tag{7.2}$$

The scheme proposed in this work achieves this bound. A coding scheme designed to tolerate s stragglers consists of an *encoding matrix* $\mathbf{B}$, and a collection of decoding vectors $\{\mathbf{a}_\mathcal{F} : \mathcal{F} \subset [n], |\mathcal{F}| = n - s\}$. The matrix $\mathbf{B}$ should satisfy:

1. Each row of $\mathbf{B}$ contains exactly w nonzero entries.
2. The linear space generated by any f rows of $\mathbf{B}$ contains the all-one vector of length k, $\mathbf{1}_{1\times k}$.

The values of these nonzero entries prescribe the linear combination sent by W_i. In other words, the *coded partial gradient* sent from W_i to M is given by

$$c_i = \sum_{j=1}^{k} \mathbf{B}_{i,j} g_j = \mathbf{B}_i g, \tag{7.3}$$

where $\mathbf{B}_i$ denotes the i^{th} row of $\mathbf{B}$. The c_i's define the encoded computation matrix $\mathbf{C} \in \mathbb{C}^{n \times p}$ as $\mathbf{C} = \begin{bmatrix} c_1^T & c_2^T & \cdots & c_n^T \end{bmatrix}^T = \mathbf{B}g$, where $c_i \in \mathbb{C}^{1 \times p}$. The decoding vectors are chosen as follows: let $\mathcal{F} = \{i_1, \ldots, i_f\}$ be the indices of the returning machines and let $\mathbf{B}_\mathcal{F}$ be the sub-matrix of $\mathbf{B}$ with rows indexed by $\mathcal{F}$. If $\mathbf{1}_{1\times k}$ is in the linear space generated by the rows of $\mathbf{B}_\mathcal{F}$, as the second property suggests, $\mathbf{a}_\mathcal{F}$ is chosen such that $\mathbf{a}_\mathcal{F}\mathbf{B}_\mathcal{F} = \mathbf{1}_{1\times k}$. As a result, we have

$$\mathbf{a}_\mathcal{F}\mathbf{C}_\mathcal{F} = \mathbf{a}_\mathcal{F}\mathbf{B}_\mathcal{F} g = \mathbf{1}_{1\times k}\ g = \nabla L(\mathcal{D}; \beta). \tag{7.4}$$

When this holds for any set of indices $\mathcal{F} \subset [n]$ of size f, it means that the gradient can be recovered from the set of f machines that return fastest.

The pseudocode listing of Algorithm 5 outlines the overall procedure for implementing our scheme, as just described.

Algorithm 5 Pseudocode of the proposed scheme

Require:
  $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$: Dataset partition
  $\mathbf{B}_1, \ldots, \mathbf{B}_n$: Encoding vectors
  T: Number of iterations
  $\{\eta_t\}_{t=1}^T$: Learning rates
Ensure: β_T: Parameters
  Partition $\mathcal{D}$ into $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$
  Assign to W_i the partitions $\{\mathcal{D}_{i_1}, \ldots \mathcal{D}_{i_w}\}$
  Assign to W_i the encoding vector $\mathbf{B}_i$
  $\beta_0 \leftarrow \mathbf{0}_{p\times 1}$
  while $t < T$ **do**
    M sends out β_t to $W_1, \ldots, W_n$
    W_i computes partial gradients $g_{i_1}, \ldots, g_{i_w}$
    W_i encodes $g_{i_1}, \ldots, g_{i_w}$ to c_i using $\mathbf{B}_i$
    M computes $\mathbf{a}_\mathcal{F}$ corresponding to first f returning machines
    M recovers $\nabla(\mathcal{D}, \beta_t)$ using $\{c_i\}_{i\in\mathcal{F}}$ and $\mathbf{a}_\mathcal{F}$
    M updates model: $\beta_{t+1} = \beta_t - \eta_t \nabla(\mathcal{D}, \beta_t)$
  end while
  return β_T

7.2.2 Computational Trade-offs

In a distributed scheme that does not employ redundancy, the taskmaster has to wait for all the workers to finish in order to compute the full gradient. However, in the scheme outlined above, the taskmaster needs to wait for the fastest f machines to recover the full gradient. Clearly, this requires more computation by each machine. Note that in the uncoded setting, the amount of computation that each worker does is $\frac{1}{n}$ of the total work, whereas in the coded setting each machine performs a $\frac{w}{k}$ fraction of the total work. From (7.2), we know that if a scheme can tolerate s stragglers, the fraction of computation that each worker does is $\frac{w}{k} \geq \frac{s+1}{n}$. Therefore, the computation load of each worker increases by a factor of $(s + 1)$. As will be explained further in Section 7.5, there is a sweet spot for $\frac{w}{k}$ (and consequently s) that minimizes the expected total time that the master waits in order to recover the full gradient update.

It is worth noting that it is often assumed [197, 123, 70] that the decoding vectors are precomputed for all possible combinations of returning machines, and the decoding cost is not taken into account in the total computation time. In a practical system, however, it is not very reasonable to compute and store all the decoding vectors, especially as there are $\binom{n}{f}$ such vectors, which grows quickly with n. In this work, we introduce an online algorithm for computing the decoding vectors on the fly, for the indices of the f workers that respond first. The approach is based on the idea of inverting Vandermonde matrices, which can be done very efficiently. In the sequel, we show how to construct an encoding matrix $\mathbf{B}$ for any w, k and n, such that the system is resilient to $\left\lfloor \frac{wn}{k} \right\rfloor - 1$ stragglers, along with an efficient algorithm for computing the decoding vectors $\{\mathbf{a}_{\mathcal{F}} : \mathcal{F} \subset [n], |\mathcal{F}| = f\}$.

7.3 Code Construction

The basic building block of our encoding scheme is a matrix $\mathbf{M} \in \{0, 1\}^{n \times k}$, where each row is of weight w, which serves as a *mask* for the matrix $\mathbf{B}$, where w is the number of data partitions that is assigned to every machine. Each column of $\mathbf{B}$ will be chosen as a codeword from a suitable Reed–Solomon Code over the complex field, with support dictated by the corresponding column in $\mathbf{M}$. Whereas the authors of [197] choose the *rows* of $\mathbf{B}$ as codewords from a suitable MDS code, this approach does not immediately work when k is not equal to n.

7.3.1 Balanced Mask Matrices

We will utilize techniques from [90, 91] to construct the matrix $\mathbf{M}$ (and then $\mathbf{B}$). For that, we present the following definition.

Definition 7 (Balanced Matrix)**.** *A matrix $\mathbf{M} \in \{0,1\}^{n\times k}$ is column (row)-balanced if for fixed row (column) weight, the weights of any two columns (rows) differ by at most 1.*

Ultimately, we are interested in a matrix $\mathbf{M}$ with row weight w that prescribes a mask for the encoding matrix $\mathbf{B}$. As an example, let $n = 8$, $k = 4$ and $w = 3$. Then, $\mathbf{M}$ is given by

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \tag{7.5}$$

where each column is of weight $\frac{nw}{k} = 6$. The following algorithm produces a balanced mask matrix. For a fixed column weight d, each row has weight either $\lfloor \frac{kd}{n} \rfloor$ or $\lceil \frac{kd}{n} \rceil$.

Algorithm 6 RowBalancedMaskMatrix(n,k,d,t)

Input:
  n: Number of rows
  k: Number of columns
  d: Weight of each column
  t: Offset parameter
Output: Row-balanced $\mathbf{M} \in \{0,1\}^{n\times k}$
  $\mathbf{M} \leftarrow \mathbf{0}_{n\times k}$
  for $j = 0$ to $k-1$ **do**
    for $i = 0$ to $d-1$ **do**
      $r = (i + jd + t)_n$ ▷ The quantity $(x)_n$ denotes x modulo n.
      $\mathbf{M}_{r,j} = 1$
    end for
  end for
  return M

As a result, when d is chosen as $\frac{nw}{k} \in \mathbb{Z}$, all rows will be of weight w. As an example, the matrix $\mathbf{M}$ in (7.5) is generated by calling RowBalancedMaskMatrix(8,4,6,0).

Algorithm 6 can be used to generate a mask matrix $\mathbf{M}$ for the encoding matrix $\mathbf{B}$: The j^{th} column of $\mathbf{B}$ will be chosen as a Reed–Solomon codeword whose support is that of the j^{th} column of $\mathbf{M}$.

7.3.2 Correctness of Algorithm 6

To lighten notation, we prove correctness for $t = 0$. The general case follows immediately.

Proposition 34. *Let k, d and n be integers where $d < n$. The row weights of matrix $\mathbf{M} \in \{0,1\}^{n\times k}$ produced by Algorithm 6 for $t = 0$ are*

$$w_i = \begin{cases} \left\lceil \frac{kd}{n} \right\rceil, & i \in \{0, \ldots, (kd-1)_n\}, \\ \left\lfloor \frac{kd}{n} \right\rfloor, & i \in \{(kd)_n, \ldots, n-1\}. \end{cases}$$

Proof. The nonzero entries in column j of $\mathbf{M}$ are given by

$$\mathcal{S}_j = \{jd, \ldots, (j+1)d - 1\}_n,$$

where the subscript n denotes reducing the elements of the set modulo n. Collectively, the nonzero indices in all columns are given by

$$\mathcal{S} = \{0, \ldots d-1, \ldots, (k-1)d, \ldots kd-1\}_n.$$

In case $n \mid kd$, each element in $\mathcal{S}$, after reducing modulo n, appears the same number of times. As a result, those indices correspond to columns of equal weight, namely $\frac{kd}{n}$. Hence, the two cases of w_i are identical along with their corresponding index sets.

In the case where $n \nmid kd$, each of the first $\left\lfloor \frac{kd}{n} \right\rfloor n$ elements, after reducing modulo n, appears the same number of times. As a result, the nonzero entries corresponding to those indices are distributed evenly amongst the n rows, each of which is of weight $\left\lfloor \frac{kd}{n} \right\rfloor$. The remaining indices $\{\left\lfloor \frac{kd}{n} \right\rfloor n, \ldots, kd-1\}_n$ contribute an additional nonzero entry to their respective rows, those indexed by $\{0, \ldots, (kd-1)_n\}$. Finally, we have that the first $(kd)_n$ rows are of weight $\left\lfloor \frac{kd}{n} \right\rfloor + 1 = \left\lceil \frac{kd}{n} \right\rceil$, while the remaining ones are of weight $\left\lfloor \frac{kd}{n} \right\rfloor$. □

Now consider the case when t is not necessarily equal to zero. This amounts to shifting (cyclically) the entries in each column by t positions downwards. As a result, the rows themselves are shifted by the same amount, allowing us to conclude the following.

Corollary 35. *Let k, d, and n be integers where $d < n$. The row weights of matrix $M \in \{0,1\}^{n\times k}$ produced by Algorithm 6 are*

$$w_i = \begin{cases} \left\lceil \frac{kd}{n} \right\rceil & i \in \{t, \ldots, (t+kd-1)_n\}, \\ \left\lfloor \frac{kd}{n} \right\rfloor & i \in \{0, t-1\} \cup \{(t+kd)_n, \ldots, n-1\}. \end{cases}$$

7.3.3 Reed–Solomon Codes

This subsection provides a quick overview of Reed–Solomon Codes. A Reed–Solomon code of length n and dimension f is a linear subspace $\mathsf{RS}[n, f]$ of $\mathbb{C}^n$ corresponding to the evaluation of polynomials of degree less than f with coefficients in $\mathbb{C}$ on a set of n distinct points $\{\alpha_1, \ldots, \alpha_n\}$, also chosen from $\mathbb{C}$. When $\alpha_i = \alpha^i$, where $\alpha \in \mathbb{C}$ is an n^{th} root of unity, the evaluations of the polynomial $t(x) = \sum_{i=0}^{f-1} t_i x^i$ on $\{1, \alpha, \ldots, \alpha^{n-1}\}$ corresponds to

$$\begin{aligned} \begin{bmatrix} t(1) \\ t(\alpha) \\ \vdots \\ t(\alpha^{n-1}) \end{bmatrix} &= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha & \cdots & \alpha^{f-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & \cdots & \alpha^{(n-1)(f-1)} \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{f-1} \end{bmatrix} \\ &= \mathbf{Gt}. \end{aligned} \tag{7.6}$$

It is well-known that any f rows of $\mathbf{G}$ form an invertible matrix, which implies that specifying any f evaluations $\{t(\alpha^{i_1}), \ldots, t(\alpha^{i_f})\}$ of a polynomial $t(x)$ of degree at most $f-1$ characterizes it. In particular, fixing $f-1$ evaluations of the polynomial to zero characterizes $t(x)$ uniquely up to scaling. This property will give us the ability to construct $\mathbf{B}$ from $\mathbf{M}$.

7.3.4 General construction

In case $d = \frac{wk}{n} \notin \mathbb{Z}$, the chosen row weight w prevents the existence of $\mathbf{M}$ where each column weight is minimal. We have to resort to Algorithm 7 that yields $\mathbf{M}$ comprised of two matrices $\mathbf{M}_h$ and $\mathbf{M}_l$ according to

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_h & \mathbf{M}_l \end{bmatrix}.$$

The matrices $\mathbf{M}_h$ and $\mathbf{M}_l$ are constructed using Algorithm 6. Each column of $\mathbf{M}_h$ has weight $d_h := \left\lceil \frac{nw}{k} \right\rceil$ and each column of $\mathbf{M}_l$ has weight $d_l := \left\lfloor \frac{nw}{k} \right\rfloor$. Note that according to (7.2), we require $d_l \geq 2$ in order to tolerate a positive number of stragglers.

Algorithm 7 Column-balanced Mask Matrix $\mathbf{M}$

Input:
n: Number of rows
k: Number of columns
w: Weight of each row
Output: Row-balanced $\mathbf{M} \in \{0,1\}^{n,k}$.
procedure MaskMatrix(n,k,w)
$k_h \leftarrow (nw)_k$
$d_h \leftarrow \left\lceil \frac{nw}{k} \right\rceil$
$k_l \leftarrow k - k_h$
$d_l \leftarrow \left\lfloor \frac{nw}{k} \right\rfloor$
$t \leftarrow (k_h d_h)_n$
$\mathbf{M}_h \leftarrow$ RowBalancedMaskMatrix$(n, k_h, d_h, 0)$
$\mathbf{M}_l \leftarrow$ RowBalancedMaskMatrix(n, k_l, d_l, t)
$\mathbf{M} \leftarrow \begin{bmatrix} \mathbf{M}_h & \mathbf{M}_l \end{bmatrix}$
return M
end procedure

The output of Algorithm 7 can now be used in Algorithm 8 instead of RowBalancedMaskMatrix to generate the appropriate mask mastrix $\mathbf{M}$.

7.3.5 Correctness of Algorithm 7

According to the algorithm, the condition $k \mid nw$ implies that $k_h = 0$ leading to $M = M_l$, which is constructed using Algorithm 6.

Moving on to the general case, the matrix $\mathbf{M}$ given by

$$\begin{bmatrix} \mathbf{M}_h & \mathbf{M}_l \end{bmatrix}$$

where each matrix is row-balanced. The particular choice of t in $\mathbf{M}_l$ aligns the "heavy" rows of $\mathbf{M}_h$ with the "light" rows of $\mathbf{M}_l$, and vice-versa. The algorithm works because the choice of parameters equates the number of heavy rows n_h of $\mathbf{M}_l$ to the number of light rows n_l of $\mathbf{M}_h$. The following lemma is useful in two ways.

Lemma 36. $\left\lfloor \frac{k_h d_h}{n} \right\rfloor + \left\lceil \frac{k_l d_l}{n} \right\rceil = \left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor = w.$

Proof. Note that the following holds:

$$\frac{k_h d_h}{n} + \frac{k_l d_l}{n} - 1 < \left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor < \frac{k_h d_h}{n} + \frac{k_l d_l}{n} + 1.$$

Furthermore, we have that

$$\begin{aligned}
\frac{k_h d_h}{n} + \frac{k_l d_l}{n} &= \frac{k_h(d_l+1)}{n} + \frac{(k-k_h)d_l}{n} && (7.7)\\
&= \frac{k_h}{n} + \frac{k d_l}{n} && (7.8)\\
&= \frac{wn - \left\lfloor \frac{wn}{k} \right\rfloor k}{n} + \frac{k d_l}{n} && (7.9)\\
&= \frac{wn - d_l k}{n} + \frac{k d_l}{n} && (7.10)\\
&= w. && (7.11)
\end{aligned}$$

We combine the two observations in one:

$$w - 1 < \left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor < w + 1,$$

and conclude that $\left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor = w$. □

We have shown the concatenation of a "heavy" row of $\mathbf{M}_h$ along with a "light" row of $\mathbf{M}_l$ results in one that is of weight w. It remains to show that the concatenation of $\mathbf{M}_h$ and $\mathbf{M}_l$ results of rows of this type only.

We will assume that $n \nmid k_h d_h$ holds. From Proposition 34, we have $n_l = n - (k_h d_h)_n$ and $n_h = (k_l d_l)_n$. We will show that the two quantities are in fact equal. Indeed, we can express n_l as

$$\begin{aligned}
n - (k_h d_h)_n &= n - k_h d_h + \left\lfloor \frac{k_h d_h}{n} \right\rfloor n\\
&= -k_h d_h + \left\lceil \frac{k_h d_h}{n} \right\rceil n\\
&= -(k-k_l)(d_l+1) + \left\lceil \frac{k_h d_h}{n} \right\rceil n\\
&= -(d_l k + k_h) + k_l d_l - \left\lfloor \frac{k_l d_l}{n} \right\rfloor n + nw\\
&= k_l d_l - \left\lfloor \frac{k_l d_l}{n} \right\rfloor n\\
&= (k_l d_l)_n.
\end{aligned}$$

Hence $n_l = n_h$ and by the choice of t, the "light" rows of $\mathbf{M}_h$ align with the "heavy" rows of $\mathbf{M}_l$, and vice-versa. Furthermore, Lemma 36 guarantees that each row of $\mathbf{M}$ is of weight $\left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor = w$. The same holds for the remaining rows, using the fact that $\lceil x \rceil + \lfloor y \rfloor = \lfloor x \rfloor + \lceil y \rceil$ when both x and y are non-integers.

7.3.6 Building the Encoding Matrix from the Mask Matrix

Once a mask matrix $\mathbf{M}$ has been determined using Algorithm 6, the encoding matrix $\mathbf{B}$ can be built by picking appropriate codewords from $\mathsf{RS}[n, f]$. Consider $\mathbf{M}$ in (7.5) and the following polynomials

$$t_1(x) = \kappa_1(x-\alpha^6)(x-\alpha^7), \tag{7.12}$$

$$t_2(x) = \kappa_2(x-\alpha^4)(x-\alpha^5), \tag{7.13}$$

$$t_3(x) = \kappa_3(x-\alpha^2)(x-\alpha^3), \tag{7.14}$$

$$t_4(x) = \kappa_4(x-1)(x-\alpha). \tag{7.15}$$

The constant κ_j is chosen such that the constant term of $t_j(x)$, i.e. $t_j(0)$, is equal to 1. The evaluations of $t_j(x)$ on $\{1, \alpha, \ldots, \alpha^7\}$ are collected in the vector $(t_j(1), t_j(\alpha), \ldots, t_j(\alpha^7))^T$ which sits as the j^{th} column of $\mathbf{B}$. The validity of this process can be confirmed using (7.6), and is generalized in Algorithm 8.

Algorithm 8 EncodingMatrix(n,k,w)

Input:
n: Number of rows
k: Number of columns
w: Row weight
α: n^{th} root of unity

Output: Row-balanced encoding matrix $\mathbf{B}$.

```
M ← RowBalancedMaskMatrix(n, k, w, 0)
B ← 0_{n×k}
for j = 0 to k − 1 do
    t_j(x) ← ∏_{r:M_{r,j}=0} (x − α^r)/(−α^r)
    for i = 0 to n − 1 do
        B_{i,j} = t_j(α^i)
    end for
end for
return B
```

Once the matrix $\mathbf{B}$ is specified, the corresponding decoding vectors required for computing the gradient at the taskmaster have to be characterized.

7.4 Efficient Online Decoding

We exploit the fact that $\mathbf{B}$ is constructed using Reed–Solomon codewords and show that each decoding vector $\mathbf{a}_{\mathcal{F}}$ can be computed in $O(f^2)$ time. Recall that the taskmaster should be able to compute the gradient from *any* f surviving machines, indexed by $\mathcal{F} \subseteq [n]$, according to (7.4). The j^{th} column of $\mathbf{B}$ is determined by

a polynomial $t_j(x) = \sum_{i=0}^{f-1} t_{j,i}x^i$ where $t_{j,0} = 1$. We can write $\mathbf{B}$ as $\mathbf{B} = \mathbf{GT}$, where $\mathbf{T} = \begin{bmatrix}\mathbf{t}_1 & \cdots & \mathbf{t}_k\end{bmatrix}$ and $\mathbf{t}_j$ is the vector of coefficients of $t_j(x)$, and $\mathbf{G}$ is the matrix given in (7.6). Now consider $\mathbf{C}_{\mathcal{F}}$, the coded partial gradients received from $\{W_i : i \in \mathcal{F}\}$. The rows of $\mathbf{B}$ corresponding to $\mathcal{F}$ are given by

$$\begin{aligned}\mathbf{B}_{\mathcal{F}} &= \mathbf{G}_{\mathcal{F}}\mathbf{T} \\ &= \begin{bmatrix} 1 & \alpha^{i_1} & \cdots & \alpha^{i_1(f-1)} \\ 1 & \alpha^{i_2} & \cdots & \alpha^{i_2(f-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{i_f} & \cdots & \alpha^{i_f(f-1)} \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \\ t_{1,1} & \cdots & t_{k,1} \\ \vdots & \ddots & \vdots \\ t_{1,f-1} & \cdots & t_{k,f-1}. \end{bmatrix}.\end{aligned}$$

We require a vector $\mathbf{a}_{\mathcal{F}}$ such that $\mathbf{a}_{\mathcal{F}}^T\mathbf{B}_{\mathcal{F}} = \mathbf{1}_{1\times k}$. This is equivalent to finding a vector $\mathbf{a}_{\mathcal{F}}$ such that

$$\mathbf{a}_{\mathcal{F}}^T\mathbf{G}_{\mathcal{F}} = (1, 0, \ldots, 0). \tag{7.16}$$

Indeed, the matrix $\mathbf{G}_{\mathcal{F}}$ in the above product is a Vandermonde matrix defined by f distinct elements and so it is invertible in $O(f^2)$ time [37], which facilitates the online computation of the decoding vectors. This is an improvement compared to previous works [197] where the decoding time is usually $O(f^3)$. Note that solving linear systems with Vandermonde matrices, in general, can be numerically unstable. However, the Vandermonde systems that we deal with here are very specific ones, i.e. their elements are all roots of unity, meaning that the Vandermonde matrix is a subset of the rows of a Fourier matrix, which has well-conditioned submatrices. A careful inspection of inverses of Vandermonde matrices built from an n^{th} root of unity allows us to compute the required decoding vector in a space efficient manner. This is demonstrated in the next subsection.

7.4.1 Space-Efficient Algorithm

Note that $\mathbf{a}_{\mathcal{F}}^T$ is nothing but the first row of the inverse of $\mathbf{G}_{\mathcal{F}}$, which can be built from a set of polynomials $\{v_1(x), \ldots, v_f(x)\}$. Let the l^{th} column of $\mathbf{G}_{\mathcal{F}}^{-1}$ be $\mathbf{v}_l = (v_{l,0}, \ldots, v_{l,f-1})^T$ and associate it with $v_l(x) = \sum_{i=0}^{f-1} v_{l,i}x^i$. The condition $\mathbf{G}_{\mathcal{F}}\mathbf{v}_l = \mathbf{e}_l$, where $\mathbf{e}_l$ is the l^{th} elementary basis vector of length f, implies that $v_l(x)$ should vanish on $\{\alpha^{i_1}, \ldots, \alpha^{i_f}\} \setminus \{\alpha^{i_l}\}$. Specifically,

$$v_l(x) = \prod_{\substack{j=1 \\ j\neq l}}^{f} \frac{x - \alpha^{i_j}}{\alpha^{i_l} - \alpha^{i_j}} \tag{7.17}$$

The first row of $\mathbf{G}_{\mathcal{F}}^{-1}$ is given by $(v_{1,0}, \ldots, v_{f,0})$, where $v_{l,0}$ is the constant term of $v_l(x)$. Indeed, we have $v_{l,0} = v_l(0)$, which can be computed in closed form according

to the following formula,

$$v_{l,0} = \prod_{\substack{j=1 \\ j \neq l}}^{f} \frac{\alpha^{i_j}}{\alpha^{i_j} - \alpha^{i_l}} = \prod_{\substack{j=1 \\ j \neq l}}^{f} (1 - \alpha^{i_l - i_j})^{-1}. \tag{7.18}$$

By choosing α as a primitive n^{th} root of unity, one is guaranteed that there are only $n-1$ distinct values of $(1-\alpha^{i_l-i_j})^{-1}$. This observation proposes that the master should precompute and store the set $\{(1-\alpha^i)^{-1}\}_{i=1}^{n-1}$, and then compute each $v_{l,0}$ by utilizing lookup operations. The following algorithm outlines this procedure.

Algorithm 9 DecodingVector($\mathcal{F}$)

Input:
$\mathcal{F}$: Ordered set of surviving machines - $\{i_1, \ldots i_f\}$
α: n^{th} root of unity
Output: Decoding vector **a** associated with $\mathcal{F}$.
a $\leftarrow \mathbf{0}_f$
for $l = 1$ to f **do**
  $\mathbf{a}_l \leftarrow \prod_{j=0, j\neq l}^{f-1} (1-\alpha^{i_l - i_j})^{-1}$
end for
return a

7.5 Analysis of Total Computation Time

In this section, we provide a theoretical model which can be used to optimize the choice of parameters that define the encoding scheme. For this purpose, we model the response time of a single computing machine as

$$T = T_{\text{delay}} + T_{\text{comp}}. \tag{7.19}$$

Here, the quantity T_{comp} is the time required for a machine to compute its portion of the gradient. This quantity is equal to $c_g \frac{Nw}{k}$, where $c_g = c_g(\ell, p)$ is a constant that indicates the time of computing the gradient for a single data point which depends on the dimension of data points, p, as well as the loss function, ℓ. The second term T_{delay} reflects the random delay incurred before the machine returns with the result of its computation. We model this delay as a Pareto distributed random variable with distribution function

$$F(t) = Pr(T_{\text{delay}} \leq t) = 1 - \left(\frac{t_0}{t}\right)^{\xi} \quad \text{for } \; t \geq t_0, \tag{7.20}$$

where the quantity t_0 can be thought of the fundamental delay of the machine, i.e. the minimum time required for a machine to return in perfect conditions. Previous

works [123, 130] model the return time of a machine as a shifted exponential random variable. We propose using this approach since the heavy-tailed nature of CPU job runtime has been observed in practice [124, 93, 94].

Let T_f denote the expected time of computing the gradient using the first f machines. As a result we have

$$T_f = \mathbb{E}[T^{(f)}_{\text{delay}}] + T_{\text{comp}} + T_{\text{dec}}(f), \tag{7.21}$$

where $T^{(f)}_{\text{delay}}$ is the f^{th} ordered statistic of T_{delay}, and $T_{\text{dec}}(f)$ is the time required at the taskmaster for decoding. Here we assume n is large and define $\alpha := \frac{w}{k}$ as the fraction of the dataset assigned to each machine. For this value of α, the number of machines required for successful recovery of the gradient is given by

$$f(\alpha) = \lceil (1-\alpha)n \rceil + 1, \tag{7.22}$$

where $\lceil x \rceil$ returns the smallest integer greater than or equal to x. We can show the following result which approximates $\mathbb{E}[T^{(f)}_{\text{delay}}]$ for large values of n.

Proposition 37. *The expected value of the f^{th} order statistic of the Pareto distribution with parameter ξ will converge as n grows, i.e.,*

$$\lim_{n\to\infty} \mathbb{E}[T^{(f)}_{delay}] = \lim_{n\to\infty} \mathbb{E}[T^{(1-\alpha)n}_{delay}] = t_0 \alpha^{-\frac{1}{\xi}}. \tag{7.23}$$

Proof. From [206], the expected value of the f^{th} ordered statistic of the Pareto distribution is:

$$\mathbb{E}[T^{(f)}_{\text{delay}}] = t_0 \frac{\Gamma(n-f+1-1/\xi)\Gamma(n+1)}{\Gamma(n-f+1)\Gamma(n+1-1/\xi)},$$

where $\Gamma(x)$ is the gamma function given by $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$. We now assume that n is large and make the standard approximation

$$\Gamma(x) \sim \sqrt{\frac{2\pi}{x}} \left(\frac{x}{e}\right)^x.$$

Furthermore, (7.2) implies that the number of machines we wait for is $f = (1-\alpha)n$, for some $\alpha < 1$ which leads to

$$\begin{aligned}\mathbb{E}[T^{(f)}_{\text{delay}}] =& t_0 \left(1 - \frac{1}{\xi(\alpha n+1)}\right)^{\alpha n+\frac{1}{2}} \\ &\times \left(1 - \frac{1}{\xi(n+1)}\right)^{-n-\frac{1}{2}} \\ &\times \left(1 - \frac{(1-\alpha)n}{n+1-\frac{1}{\xi}}\right)^{-1/\xi}.\end{aligned}$$

By letting $n \to \infty$, the first two terms in the product converge to $e^{-\xi}$ and e^{ξ}, respectively, which yields

$$\lim_{n\to\infty} \mathbb{E}[T_{\text{delay}}^{(f)}] = t_0 \alpha^{-\frac{1}{\xi}}.$$

□

Using this result, we can approximate T_f, for $n \gg 1$,

$$T_f \approx t_0 \alpha^{-\frac{1}{\xi}} + c_g N\alpha + c_m(1-\alpha)^2 n^2, \tag{7.24}$$

where we assume that the taskmaster uses Algorithm 9 for decoding. If we assume c_m is the time required for one FLOP, the total decoding time is given by $c_m(f-1)f \approx c_m(1-\alpha)^2 n^2$. Since α is bounded from above by the memory of each machine, one can find the optimal computation time, subject to memory constraints, by minimizing T_f with respect to α.

7.5.1 Offline Decoding

In the schemes where the decoding vectors are computed offline, the quantity T_{dec} does not appear in the total computation time T_f. Therefore, for large values of n, we can write:

$$T_f = t_0 \alpha^{-\frac{1}{\xi}} + c_g N\alpha. \tag{7.25}$$

This function can be minimized with respect to α by standard calculus to give

$$\alpha^* = \left(\frac{t_0}{c_g N \xi}\right)^{\frac{\xi}{1+\xi}}. \tag{7.26}$$

Note that this quantity is valid (less than one) if and only if one has $\frac{t_0}{c_g N\xi} < 1$. It has been observed in practice that the parameter ξ is close to one. Therefore, this assumption holds because N is assumed to be large.

For illustrative purposes, we plot the function T_f from (7.25) for a given set of parameters and indicate the optimal point. This plot is given in Figure 7.2.

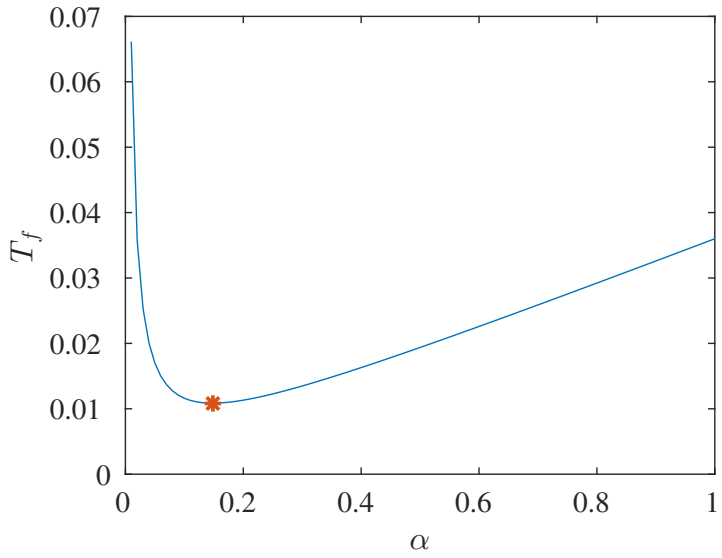


Figure 7.2: This plot corresponds to a setup where the number of training examples is $N = 12000$ and $c_g = 3 \times 10^{-6}$ to give $Nc_g = 0.035$. The parameters of the Pareto distribution corresponding to the delay is characterized by $t_0 = 0.001$ and $\xi = 1.1$. The optimizer of this function as predicted by (7.26) is $\alpha^* = 0.1477$. This point is denote by the star symbol.

7.6 Numerical Results

To demonstrate the effectiveness of the scheme, we performed numerical simulations on a simple learning task, with realistic delays. We train a softmax regression model on a distributed cluster composed of $n = 80$ processors to classify 10000 handwritten digits from the MNIST dataset [121], while synthetically introducing computation delays according to a model adopted from the literature. The delay model (and its parameters), adopted from [124, 93, 94], has a Pareto distribution (7.20) with parameters $\xi = 1.1$ and $t_0 = 0.001$.

We compare the proposed Reed–Solomon scheme (Coded - RS) with the following schemes, by running each of them on the same dataset for a fixed amount of time (in seconds), and then measuring the test error.

- Uncoded - Wait for all: Data is distributed equally amongst n machines - Wait for all n machines to return.
- Coded - MDS: Scheme described in [197].
- Uncoded - Wait for f_{RS}: Data is distributed equally amongst n machines - Wait for f_{RS} machines.

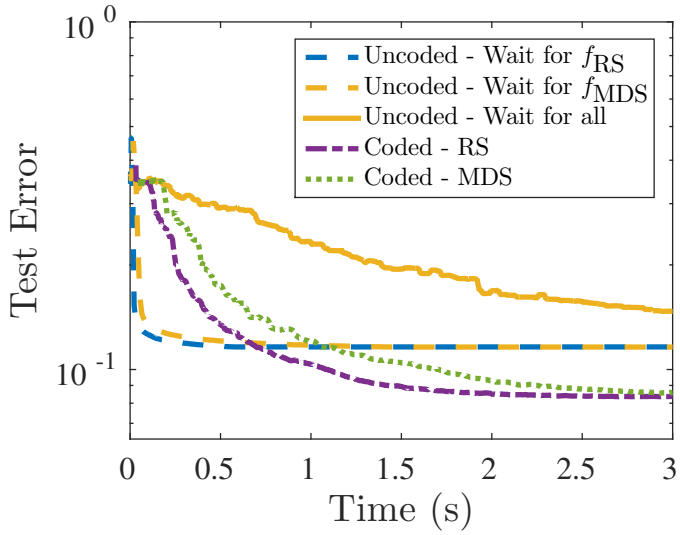


Figure 7.3: The comparison between the test error of different schemes as a function of time, for a softmax regression model trained using distributed gradient descent on $n = 80$ machines. The model was trained on 12000 examples from the MNIST database [121] and validated on a test set of size 10000. The Reed–Solomon based scheme (Coded - RS) waits for $f_{\text{RS}} = 68$ machines, while the one corresponding to [197] (Coded - MDS) waits for $f_{\text{MDS}} = 33$. f_{RS} and f_{MDS} were obtained by numerically optimizing (7.21). The two coded schemes outperform the uncoded ones. Coded-RS denotes the proposed scheme.

- Uncoded - Wait for f_{MDS}: Data is distributed equally amongst n machines - Wait for f_{MDS} machines.

Similar to [197], the knowledge of the entire gradient allows us to employ accelerated gradient methods such as the one proposed by Nesterov [152]. Details of the experiment are given in the accompanying description of Figure 7.3.

The Reed–Solomon based scheme (Coded - RS) waits for $f_{\text{RS}} = 68$ machines, while the one corresponding to [197] (Coded - MDS) waits for $f_{\text{MDS}} = 33$. These quantities were obtained by numerically optimizing the expected total computation time, mentioned in (7.21).

It is worth mentioning that our experiments show that, even on this relatively small dataset, a distributed coded solution outperforms the scenario where the computation is performed on a single machine. However, the single-machine scenario is not very interesting, as we mostly care about cases where performing the computation on one machine is infeasible.

7.7 Conclusion

We presented a straggler mitigation scheme that facilitates the implementation of distributed gradient descent in a computing cluster. For a fixed per-machine computational effort, the taskmaster recovers the full gradient from the least number of machines theoretically required, which is done via an algorithm that is efficient in both space and time. Furthermore, we propose a theoretical delay model based on heavy-tailed distributions and incorporates the decoding time, which allows us to minimize the expected running time of the algorithm.

Part IV

Learning from Data

Chapter 8

MINIMAX OPTIMALITY AND IMPLICIT REGULARIZATION OF STOCHASTIC GRADIENT/MIRROR DESCENT

[1] Navid Azizan and Babak Hassibi. "Stochastic gradient/mirror descent: Minimax optimality and implicit regularization". In: *2018 Neural Information Processing Systems (NeurIPS) Deep Learning Theory Workshop*. 2018.

[2] Navid Azizan and Babak Hassibi. "Stochastic gradient/mirror descent: Minimax optimality and implicit regularization". In: *2019 International Conference on Learning Representations (ICLR)*. 2019.

Stochastic descent methods (of the gradient and mirror varieties) have become increasingly popular in optimization. In fact, it is now widely recognized that the success of deep learning is not only due to the special deep architecture of the models, but also due to the behavior of the stochastic descent methods used, which play a key role in reaching "good" solutions that generalize well to unseen data. In an attempt to shed some light on why this is the case, we revisit some minimax properties of stochastic gradient descent (SGD) for the square loss of linear models—originally developed in the 1990s—and extend them to *general* stochastic mirror descent (SMD) algorithms for *general* loss functions and *nonlinear* models. In particular, we show that there is a fundamental identity which holds for SMD (and SGD) under very general conditions, and which implies the minimax optimality of SMD (and SGD) for sufficiently small step size, and for a general class of loss functions and general nonlinear models. We further show that this identity can be used to naturally establish other properties of SMD (and SGD), namely convergence and *implicit regularization* for over-parameterized linear models (in what is now being called the "interpolating regime"), some of which have been shown in certain cases in prior literature.

8.1 Introduction

Deep learning has proven to be extremely successful in a wide variety of tasks [118, 119, 143, 189, 213]. Despite its tremendous success, the reasons behind the good generalization properties of these methods to unseen data is not fully understood (and, arguably, remains somewhat of a mystery to this day). Initially, this success was mostly attributed to the special deep architecture of these models. However, in the past few years, it has been widely noted that the architecture is only part

of the story, and, in fact, the optimization algorithms used to train these models, typically stochastic gradient descent (SGD) and its variants, play a key role in learning parameters that generalize well.

In particular, it has been observed that since these deep models are *highly over-parameterized*, they have a lot of capacity, and can fit to virtually any (even random) set of data points [224]. In other words, highly over-parameterized models can "interpolate" the data, so much so that this regime has been called the "interpolating regime" [138]. In fact, on a given dataset, the loss function often has (uncountably infinitely) many *global* minima, which can have drastically different generalization properties, and it is not hard to construct "trivial" global minima that do not generalize. Which minimum among all the possible minima we pick in practice is determined by the optimization algorithm that we use for training the model. Even though it may seem at first that, because of the non-convexity of the loss function, the stochastic descent algorithms may get stuck in local minima or saddle points, in practice they almost always achieve a global minimum [112, 224, 122], which perhaps can also be justified by the fact that these models are highly over-parameterized. What is even more interesting is that not only do these stochastic descent algorithms converge to global minima, but they converge to "special" ones that generalize well, even in the absence of any explicit regularization or early stopping [224]. Furthermore, it has been observed that even among the common optimization algorithms, namely SGD or its variants (AdaGrad [68], RMSProp [199], Adam [114], etc.), there is a discrepancy in the solutions achieved by different algorithms and their generalization capabilities [211], which again highlights the important role of the optimization algorithm in generalization.

There have been many attempts in recent years to explain the behavior and properties of these stochastic optimization algorithms, and many interesting insights have been obtained [3, 58, 188, 193]. In particular, it has been argued that the optimization algorithms perform an *implicit regularization* [154, 137, 87, 85, 194, 86] while optimizing the loss function, which is perhaps why the solution generalizes well. Despite this recent progress, most results explaining the behavior of the optimization algorithm, even for SGD, are limited to linear or very simplistic models. Therefore, a general characterization of the behavior of stochastic descent algorithms for more general models would be of great interest.

8.1.1 Our Contribution

In this chapter, we present an alternative explanation of the behavior of SGD, and more generally, the stochastic mirror descent (SMD) family of algorithms, which includes SGD as a special case. We do so by obtaining a fundamental identity for such algorithms (see Lemmas 39 and 42). Using these identities, we show that for general nonlinear models and general loss functions, when the step size is sufficiently small, SMD (and therefore also SGD) is the optimal solution of a certain minimax filtering (or online learning) problem. The minimax formulation is inspired by, and rooted in, H^∞ filtering theory, which was originally developed in the 1990s in the context of robust control theory [98, 190, 97], and we generalize several results from this literature, e.g., [96, 115]. Furthermore, we show that many properties recently proven in the learning/optimization literature, such as the implicit regularization of SMD in the over-parameterized linear case—when convergence happens—[85], naturally follow from this theory. The theory also allows us to establish new results, such as the convergence (in a deterministic sense) of SMD in the over-parameterized linear case. We also use the theory developed in this chapter to provide some speculative arguments into why SMD (and SGD) may have similar convergence and implicit regularization properties in the so-called "highly over-parameterized" nonlinear setting (where the number of parameters far exceeds the number of data points) common to deep learning.

In an attempt to make the chapter easier to follow, we first describe the main ideas and results in a simpler setting, namely, SGD on the square loss of linear models, in Section 8.3, and mention the connections to H^∞ theory. The full results, for SMD on a general class of loss functions and for general nonlinear models, are presented in Section 8.4. We demonstrate some implications of this theory, such as deterministic convergence and implicit regularization, in Section 8.5, and we finally conclude with some remarks in Section 8.6. Most of the formal proofs are relegated to the appendix.

8.2 Preliminaries

Denote the training dataset by $\{(x_i, y_i) : i = 1, \dots, n\}$, where $x_i \in \mathbb{R}^d$ are the inputs, and $y_i \in \mathbb{R}$ are the labels. We assume that the data is generated through a (possibly nonlinear) model $f_i(w) = f(x_i, w)$ with some parameter vector $w \in \mathbb{R}^m$, plus some noise v_i, i.e., $y_i = f(x_i, w) + v_i$ for $i = 1, \dots, n$. The noise can be due to actual measurement error, or it can be due to modeling error (if the model $f(x_i, \cdot)$ is not rich enough to fully represent the data), or it can be a combination of both. As a result, we do not make any assumptions on the noise (such as stationarity, whiteness,

Gaussianity, etc.).

Since typical deep models have a lot of capacity and are highly over-parameterized, we are particularly interested in the over-parameterized (so-caled interpolating) regime, i.e., when $m > n$. In this case, there are many parameter vectors w (in fact, uncountably infinitely many) that are consistent with the observations. We denote the set of these parameter vectors by

$$\mathcal{W} = \{w \in \mathbb{R}^m \mid y_i = f(x_i, w),\ i = 1, \ldots, n\}\,. \tag{8.1}$$

(Note the absence of the noise term, since in this regime we can fully interpolate the data.) The set $\mathcal{W}$ is typically an $(m-n)$-dimensional manifold and depends only on the training data $\{(x_i, y_i) : i = 1, \ldots, n\}$ and nonlinear model $f(\cdot, \cdot)$.

The total loss on the training set (empirical risk) can be denoted by $L(w) = \sum_{i=1}^{n} L_i(w)$, where $L_i(\cdot)$ is the loss on the individual data point i. We assume that the loss $L_i(\cdot)$ depends only on the residual, i.e., the difference between the prediction and the true label. In other words,

$$L_i(w) = l(y_i - f(x_i, w)), \tag{8.2}$$

where $l(\cdot)$ can be any nonnegative differentiable function with $l(0) = 0$. Typical examples of $l(\cdot)$ include square (l_2) loss, Huber loss, etc. We remark that, in the interpolating regime, every parameter vector in the set $\mathcal{W}$ renders each individual loss zero, i.e., $L_i(w) = 0$, for all $w \in \mathcal{W}$.

8.3 Warm-up: Revisiting SGD on Square Loss of Linear Models

In this section, we describe the main ideas and results in a simple setting, i.e., stochastic gradient descent (SGD) for the square loss of a linear model, and we revisit some of the results from H^∞ theory [98, 190]. In this case, the data model is $y_i = x_i^T w + v_i$, $i = 1, \ldots, n$ (where there is no assumption on v_i), and the loss function is $L_i(w) = \frac{1}{2}(y_i - x_i^T w)^2$.

Assuming the data is indexed randomly, the SGD updates are defined as $w_i = w_{i-1} - \eta \nabla L_i(w_{i-1})$, where $\eta > 0$ is the step size or learning rate.[1] The update in this case can be expressed as

$$w_i = w_{i-1} + \eta \left(y_i - x_i^T w_{i-1}\right) x_i, \tag{8.3}$$

for $i \geq 1$ (for $i > n$, we can either cycle through the data, or select them at random).

[1] For the sake of simplicity of presentation, we present the results for constant step size. We show in the appendix that all the results extend to the case of time-varying step-size.

Remark. *We should point out that, when the step size η is fixed, the SGD recursions have no hope of converging, unless there exists a weight vector w which perfectly interpolates the data $\{(x_i, y_i) : i = 1, \ldots, n\}$. The reason being that, if this is not the case, for any estimated weight vector in SGD, there will exist at least one data point that has a nonzero instantaneous gradient and that will therefore move the estimate by a non-vanishing amount.*[2] *It is for this reason that the results on the convergence of SGD and SMD (Sections 8.3.3 and 8.5) pertain to the interpolating regime.*

8.3.1 Conservation of Uncertainty

Prior to the i-th step of any optimization algorithm, we have two sources of uncertainty: our uncertainty about the unknown parameter vector w, which we can represent by $w - w_{i-1}$, and our uncertainty about the i-th data point (x_i, y_i), which we can represent by the noise v_i. After the i-th step, the uncertainty about w is transformed to $w - w_i$. But what about the uncertainty in v_i? What is it transformed to? In fact, we will view any optimization algorithm as one which redistributes the uncertainties at time $i - 1$ to new uncertainties at time i. The two uncertainties, or error terms, we will consider are e_i and $e_{p,i}$, defined as follows.

$$e_i := y_i - x_i^T w_{i-1}, \text{ and } e_{p,i} := x_i^T w - x_i^T w_{i-1}. \tag{8.4}$$

e_i is often referred to as the *innvovations* and is the error in predicting y_i, given the input x_i. $e_{p,i}$ is sometimes called the *prediction error*, since it is the error in predicting the noiseless output $x_i^T w$, i.e., in predicting what the best output of the model is. In the absence of noise, e_i and $e_{p,i}$ coincide.

One can show that SGD transforms the uncertainties in the fashion specified by the following lemma, which was first noted in [97].

Lemma 38. *For any parameter w and noise values $\{v_i\}$ that satisfy $y_i = x_i^T w + v_i$ for $i = 1, \ldots, n$, and for any step size $\eta > 0$, the following relation holds for the SGD iterates $\{w_i\}$ given in Eq.* (8.3)

$$\|w - w_{i-1}\|^2 + \eta v_i^2 = \|w - w_i\|^2 + \eta \left(1 - \eta \|x_i\|^2\right) e_i^2 + \eta e_{p,i}^2, \quad \forall i \geq 1. \tag{8.5}$$

As illustrated in Figure 8.1, this means that each step of SGD can be thought of as a lossless transformation of the input uncertainties to the output uncertainties, with the specified coefficients.

[2] Of course, one may get convergence by having a vanishing step size $\eta_i \to 0$. However, in this case, convergence is not surprising—since, effectively, after a while, the weights are no longer being updated—and the more interesting question is "what" the recursion converges to.

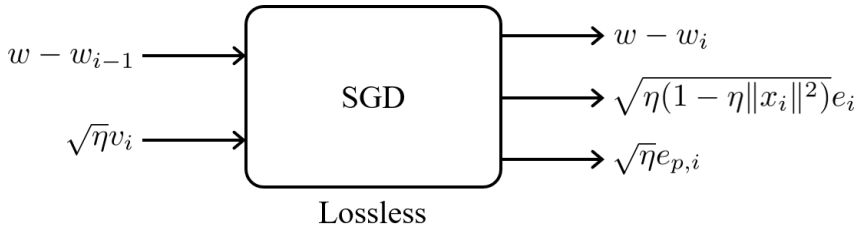


Figure 8.1: Illustration of Lemma 38. Each step of SGD can be viewed as a transformation of the uncertainties with the right coefficients.

Once one knows this result, proving it is straightforward. To see that, note that we can write $v_i = y_i - x_i^T w$ as $v_i = (y_i - x_i^T w_{i-1}) - (x_i^T w - x_i^T w_{i-1})$. Multiplying both sides by $\sqrt{\eta}$, we have

$$\sqrt{\eta} v_i = \sqrt{\eta}(y_i - x_i^T w_{i-1}) - \sqrt{\eta}(x_i^T w - x_i^T w_{i-1}). \tag{8.6}$$

On the other hand, subtracting both sides of the update rule (8.3) from w yields

$$w - w_i = (w - w_{i-1}) - \eta \left(y_i - x_i^T w_{i-1}\right) x_i. \tag{8.7}$$

Squaring both sides of (8.6) and (8.7), and subtracting the results leads to Equation (8.5).

A nice property of Equation (8.5) is that, if we sum over all $i = 1, \ldots, T$, the terms $\|w - w_i\|^2$ and $\|w - w_{i-1}\|^2$ on different sides cancel out telescopically, leading to the following important lemma.

Lemma 39. *For any parameter w and noise values $\{v_i\}$ that satisfy $y_i = x_i^T w + v_i$ for $i = 1, \ldots, n$, any initialization w_0, any step size $\eta > 0$, and any number of steps $T \geq 1$, the following relation holds for the SGD iterates $\{w_i\}$ given in Eq.* (8.3)

$$\boxed{\|w - w_0\|^2 + \eta \sum_{i=1}^{T} v_i^2 = \|w - w_T\|^2 + \eta \sum_{i=1}^{T} \left(1 - \eta\|x_i\|^2\right) e_i^2 + \eta \sum_{i=1}^{T} e_{p,i}^2.} \tag{8.8}$$

As we will show next, this identity captures most properties of SGD, and implies several important results in a very transparent fashion. For this reason, this relation can be viewed as a "fundamental identity" for SGD.

8.3.2 Minimax Optimality of SGD

For a given horizon T, consider the following minimax problem:

$$\min_{\{w_i\}} \max_{w, \{v_i\}} \frac{\|w - w_T\|^2 + \eta \sum_{i=1}^{T} e_{p,i}^2}{\|w - w_0\|^2 + \eta \sum_{i=1}^{T} v_i^2}. \tag{8.9}$$

This minimax problem is motivated by the theory of H^∞ control and estimation [74, 98, 31]. The denominator of the cost function can be interpreted as the *energy of the uncertainties* and consists of two terms, $\|w - w_0\|^2$, the energy of our uncertainty of the unknown weight vector at the beginning of learning when we have not yet observed the data, and $\sum_{i=1}^T v_i^2$, the energy of the uncertainty in the measurements. The numerator denotes the energy of the estimation errors in an *online setting*. The first term, $\|w - w_T\|^2$, is the energy of our uncertainty of the unknown weight vector after we have observed T data points, and the second term, $\sum_{i=1}^T e_{p,i}^2 = \sum_{i=1}^T (x_i^T w - x_i^T w_{i-1})^2$, is the energy of the prediction error, i.e., how well we can predict the true uncorrupted output $x_i^T w$ using measurements up to time $i-1$. The parameter η weighs the two energy terms relative to each other. In this minimax problem, nature has access to the unknown weight vector w and the noise sequence v_i and would like to maximize the energy gain from the uncertainties to prediction errors (so that the estimator behaves poorly), whereas the estimator attempts to minimize the energy gain. Such an estimator is referred to as H^∞-optimal and is robust because it safeguards against the worst-case noise. It is also conservative, for the exact same reason.[3]

Theorem 40. *For any initialization w_0, any step size $0 < \eta \leq \min_i \frac{1}{\|x_i\|^2}$, and any number of steps $T \geq 1$, the stochastic gradient descent iterates $\{w_i\}$ given in Eq.* (8.3) *are the optimal solution to the minimax problem (8.9). Furthermore, the optimal minimax value (achieved by SGD) is* 1.

This theorem explains the observed robustness and conservatism of SGD. Despite the conservativeness of safeguarding against the worst-case disturbance, this choice may actually be the rational thing to do in situations where we do not have much knowledge about the disturbances, which is the case in many machine learning tasks.

Theorem 40 holds for any horizon $T \geq 1$. A variation of this result, i.e., when $T \to \infty$ and without the $\|w - w_T\|^2$ term in the numerator, was first shown in [96, 97]. In that case, the ratio $\frac{\eta \sum_{i=1}^\infty e_{p,i}^2}{\|w-w_0\|^2 + \eta \sum_{i=1}^\infty v_i^2}$ in the minimax problem is in fact the H^∞ *norm* of the transfer operator that maps the unknown disturbances $(w - w_0, \{\sqrt{\eta} v_i\})$ to the prediction errors $\{\sqrt{\eta} e_{p,i}\}$.

[3] The setting described is somewhat similar to the setting of online learning, where one considers the relative performance of an online learner who needs to predict, compared to a clairvoyant one who has access to the entire data set [183, 99]. In online learning, the relative performance is described as a difference, rather than as a ratio in H^∞ theory, and is referred to as *regret*.

We end this section with a stochastic interpretation of SGD [97]. Assume that the true weight vector has a normal distribution with mean w_0 and covariance matrix ηI, and that the noise v_i are iid standard normal. Then SGD solves

$$\min_{\{w_i\}} \mathbb{E} \exp\left(\frac{1}{2} \cdot \left(\|w - w_T\|^2 + \eta \sum_{i=1}^{T} (x_i^T w - x_i^T w_{i-1})^2\right)\right), \tag{8.10}$$

and no exponent larger than $\frac{1}{2}$ is possible, in the sense that no estimator can keep the expected cost finite. This means that, in the Gaussian setting, SGD minimizes the expected value of an *exponential* quadratic cost. The algorithm is thus very adverse to large estimation errors, as they are penalized exponentially larger than moderate ones.

8.3.3 Convergence and Implicit Regularization

The over-parameterized (interpolating) linear regression regime is a simple but instructive setting, recently considered in some papers [85, 224]. In this setting, we can show that, for sufficiently small step, i.e., $0 < \eta \leq \min_i \frac{1}{\|x_i\|^2}$, SGD always converges to a special solution among all the solutions $\mathcal{W}$, in particular to the one with the smallest l_2 distance from w_0. In other words, if, for example, initialized at zero, SGD implicitly regularizes the solution according to an l_2 norm. This result follows directly from Lemma 39.

To see that, note that in the interpolating case the v_i are zero, and we have $e_i = y_i - x_i^T w_{i-1} = x_i^T w - x_i^T w_{i-1} = e_{p,i}$. Hence, identity (8.8) reduces to

$$\|w - w_0\|^2 = \|w - w_T\|^2 + \eta \sum_{i=1}^{T} \left(2 - \eta \|x_i\|^2\right) e_i^2, \tag{8.11}$$

for all $w \in \mathcal{W}$. By dropping the $\|w - w_T\|^2$ term and taking $T \to \infty$, we have $\eta \sum_{i=1}^{\infty} \left(2 - \eta \|x_i\|^2\right) e_i^2 \leq \|w - w_0\|^2$, which implies that, for $0 < \eta < \min_i \frac{2}{\|x_i\|^2}$, we must have $e_i \to 0$ as $i \to \infty$. When $e_i = y_i - x_i^T w_{i-1}$ goes to zero, the updates in (8.3) vanish and we get convergence, i.e., $w \to w_\infty$. Further, again because $e_i \to 0$, all the data points are being fit, which means $w_\infty \in \mathcal{W}$. Moreover, it is again very straightforward to see from (8.11) that the solution converged to is the one with minimum Euclidean norm from the initial point. To see that, notice that the summation term in Eq. (8.11) is *independent of* w (it depends only on x_i, y_i, and w_0). Therefore, by taking $T \to \infty$ and minimizing both sides with respect to $w \in \mathcal{W}$, we get

$$w_\infty = \arg\min_{w \in \mathcal{W}} \|w - w_0\|. \tag{8.12}$$

Once again, this also implies that if SGD is initialized at the origin, i.e., $w_0 = 0$, then it converges to the minimum-l_2-norm solution, among all the solutions.

8.4 Main Result: General Characterization of Stochastic Mirror Descent

Stochastic Mirror Descent (SMD) [149, 33, 55, 227] is one of the most widely used families of algorithms for stochastic optimization, which includes SGD as a special case. In this section, we provide a characterization of the behavior of general SMD, on *general* loss functions and *general* nonlinear models, in terms of a fundamental identity and minimax optimality.

For any strictly convex and differentiable potential $\psi(\cdot)$, the corresponding SMD updates are defined as

$$w_i = \arg\min_{w} \ \eta w^T \nabla L_i(w_{i-1}) + D_\psi(w, w_{i-1}), \tag{8.13}$$

where

$$D_\psi(w, w_{i-1}) = \psi(w) - \psi(w_{i-1}) - \nabla\psi(w_{i-1})^T(w - w_{i-1}) \tag{8.14}$$

is the Bregman divergence with respect to the potential function $\psi(\cdot)$. Note that $D_\psi(\cdot,\cdot)$ is non-negative, convex in its first argument, and that, due to strict convexity, $D_\psi(w, w') = 0$ iff $w = w'$. Moreover, the updates can be equivalently written as

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta\nabla L_i(w_{i-1}), \tag{8.15}$$

which are uniquely defined because of the invertibility of $\nabla\psi$ (again, implied by the strict convexity of $\psi(\cdot)$). In other words, stochastic mirror descent can be thought of as transforming the variable w, with a *mirror map* $\nabla\psi(\cdot)$, and performing the SGD update on the new variable. For this reason, $\nabla\psi(w)$ is often referred to as the *dual* variable, while w is the *primal* variable.

Different choices of the potential function $\psi(\cdot)$ yield different optimization algorithms, which, as we will see, result in different implicit regularizations. To name a few examples: For the potential function $\psi(w) = \frac{1}{2}\|w\|^2$, the Bregman divergence is $D_\psi(w, w') = \frac{1}{2}\|w - w'\|^2$, and the update rule reduces to that of SGD. For $\psi(w) = \sum_j w_j \log w_j$, the Bregman divergence becomes the unnormalized relative entropy (Kullback-Leibler divergence) $D_\psi(w, w') = \sum_j w_j \log \frac{w_j}{w'_j} - \sum_j w_j + \sum_j w'_j$, which corresponds to the exponentiated gradient descent (or the exponential weights) algorithm. Other examples include $\psi(w) = \frac{1}{2}\|w\|_Q^2 = \frac{1}{2}w^T Q w$ for a positive definite matrix Q, which yields $D_\psi(w, w') = \frac{1}{2}(w - w')^T Q(w - w')$, and the q-norm squared $\psi(w) = \frac{1}{2}\|w\|_q^2$, which with $\frac{1}{p} + \frac{1}{q} = 1$ yields the p-norm algorithms [82, 78].

In order to derive an equivalent "conservation law" for SMD, similar to the identity (8.5), we first need to define a new measure for the difference between the parameter vectors w and w' according to the loss function $L_i(\cdot)$. To that end, let us define

$$D_{L_i}(w, w') := L_i(w) - L_i(w') - \nabla L_i(w')^T (w - w'), \tag{8.16}$$

which is defined in a similar way to a Bregman divergence for the loss function.[4] The difference though is that, unlike the potential function of the Bregman divergence, the loss function $L_i(\cdot) = \ell(y_i - f(x_i, \cdot))$ need not be convex, even when $\ell(\cdot)$ is, due to the nonlinearity of $f(\cdot, \cdot)$. As a result, $D_{L_i}(w, w')$ is not necessarily non-negative. The following result, which is the general counterpart of Lemma 38, states the identity that characterizes SMD updates in the general setting.

Lemma 41. *For any (nonlinear) model $f(\cdot, \cdot)$, any differentiable loss $l(\cdot)$, any parameter w and noise values $\{v_i\}$ that satisfy $y_i = f(x_i, w) + v_i$ for $i = 1, \dots, n$, and any step size $\eta > 0$, the following relation holds for the SMD iterates $\{w_i\}$ given in Eq.* (8.15)

$$D_\psi(w, w_{i-1}) + \eta l(v_i) = D_\psi(w, w_i) + E_i(w_i, w_{i-1}) + \eta D_{L_i}(w, w_{i-1}), \tag{8.17}$$

for all $i \geq 1$, where

$$E_i(w_i, w_{i-1}) := D_\psi(w_i, w_{i-1}) - \eta D_{L_i}(w_i, w_{i-1}) + \eta L_i(w_i). \tag{8.18}$$

The proof is provided in Appendix 8.A. Note that $E_i(w_i, w_{i-1})$ is not a function of w. Furthermore, even though it does not have to be nonnegative in general, for η sufficiently small, it becomes nonnegative, because the Bregman divergence $D_\psi(.,.)$ is nonnegative.

Summing Equation (8.17) over all $i = 1, \dots, T$ leads to the following identity, which is the general counterpart of Lemma 39.

Lemma 42. *For any (nonlinear) model $f(\cdot, \cdot)$, any differentiable loss $l(\cdot)$, any parameter w and noise values $\{v_i\}$ that satisfy $y_i = f(x_i, w) + v_i$ for $i = 1, \dots, n$, any initialization w_0, any step size $\eta > 0$, and any number of steps $T \geq 1$, the following*

[4] It is easy to verify that for linear models and quadratic loss we obtain $D_{L_i}(w, w') = (x_i^T w - x_i^T w')^2$.

relation holds for the SMD iterates $\{w_i\}$ *given in Eq.* (8.15)

$$\boxed{D_\psi(w, w_0) + \eta \sum_{i=1}^{T} l(v_i) = D_\psi(w, w_T) + \sum_{i=1}^{T} \left(E_i(w_i, w_{i-1}) + \eta D_{L_i}(w, w_{i-1}) \right).}$$

(8.19)

We should reiterate that Lemma 42 is a fundamental property of SMD, which allows one to prove many important results, in a direct way.

In particular, in this setting, we can show that SMD is minimax optimal in a manner that generalizes Theorem 40 of Section 8.3, in the following 3 ways: 1) General potential $\psi(\cdot)$, 2) General model $f(\cdot, \cdot)$, and 3) General loss function $l(\cdot)$. The result is as follows.

Theorem 43. *Consider any (nonlinear) model* $f(\cdot, \cdot)$*, any non-negative differentiable loss* $l(\cdot)$ *with the property* $l(0) = l'(0) = 0$*, and any initialization* w_0*. For sufficiently small step size, i.e., for any* $\eta > 0$ *for which* $\psi(w) - \eta L_i(w)$ *is convex for all i, and for any number of steps* $T \geq 1$*, the SMD iterates* $\{w_i\}$ *given by Eq.* (8.15)*, w.r.t. any strictly convex potential* $\psi(\cdot)$*, is the optimal solution to the following minimization problem*

$$\min_{\{w_i\}} \max_{w, \{v_i\}} \frac{D_\psi(w, w_T) + \eta \sum_{i=1}^{T} D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \eta \sum_{i=1}^{T} l(v_i)}. \tag{8.20}$$

Furthermore, the optimal value (achieved by SMD) is 1.

The proof is provided in Appendix 8.B. For the case of square loss and a linear model, the result reduces to the following form.

Corollary 44. *For* $L_i(w) = \frac{1}{2}(y_i - x_i^T w)^2$*, for any initialization* w_0*, any sufficiently small step size, i.e.,* $0 < \eta \leq \frac{\alpha}{\|x_i\|^2}$*, and any number of steps* $T \geq 1$*, the SMD iterates* $\{w_i\}$ *given by Eq.* (8.15)*, w.r.t. any* α*-strongly convex potential* $\psi(\cdot)$*, is the optimal solution to*

$$\min_{\{w_i\}} \max_{w, \{v_i\}} \frac{D_\psi(w, w_T) + \frac{\eta}{2} \sum_{i=1}^{T} e_{p,i}^2}{D_\psi(w, w_0) + \frac{\eta}{2} \sum_{i=1}^{T} v_i^2}. \tag{8.21}$$

The optimal value (achieved by SMD) is 1.

We should remark that Theorem 43 and Corollary 44 generalize several known results in the literature. In particular, as mentioned in Section 8.3, the result of [96] is a

special case of Corollary 44 for $\psi(w) = \frac{1}{2}\|w\|^2$. Furthermore, our result generalizes the result of [115], which is the special case for the p-norm algorithms, again, with square loss and a linear model. Another interesting connection to the literature is that it was shown in [95] that SGD is *locally* minimax optimal, with respect to the H^∞ norm. Strictly speaking, our result is not a generalization of that result; however, Theorem 43 can be interpreted as SGD/SMD being *globally* minimax optimal, but with respect to different metrics in the numerator and denominator. Namely, the uncertainty about the weight vector w is measured by the Bregman divergence of the potential, the uncertainty about the noise by the loss, and the prediction error by the "Bregman-divergence-like" expression of the loss.

8.5 Convergence and Implicit Regularization in Over-Parameterized Models

In this section, we show some of the implications of the theory developed in the previous section. In particular, we show convergence and implicit regularization, in the over-parameterized (so-called interpolating) regime[5], for general SMD algorithms. We first consider the linear interpolating case, which has been studied in the literature, and show that the known results follow naturally from our Lemma 42. Further, we shall obtain some *new* convergence results.

8.5.1 Over-Parameterized Linear Models

In this setting, the v_i are zero, $\mathcal{W} = \left\{w \mid y_i = x_i^T w,\ i = 1, \dots, n\right\}$, and $L_i(w) = l(y_i - x_i^T w)$, with any differentiable loss $l(\cdot)$. Therefore, Eq. (8.19) reduces to

$$D_\psi(w, w_0) = D_\psi(w, w_T) + \sum_{i=1}^{T} \left(E_i(w_i, w_{i-1}) + \eta D_{L_i}(w, w_{i-1})\right), \tag{8.22}$$

for all $w \in \mathcal{W}$, where

$$\begin{aligned} D_{L_i}(w, w_{i-1}) &= L_i(w) - L_i(w_{i-1}) - \nabla L_i(w_{i-1})^T (w - w_{i-1}) && (8.23)\\ &= 0 - l(y_i - x_i^T w_{i-1}) + l'(y_i - x_i^T w_{i-1}) x_i^T (w - w_{i-1}) && (8.24)\\ &= -l(y_i - x_i^T w_{i-1}) + l'(y_i - x_i^T w_{i-1})(y_i - x_i^T w_{i-1}) && (8.25) \end{aligned}$$

which is notably *independent of* w. As a result, we can easily minimize both sides of Eq. (8.22) with respect to $w \in \mathcal{W}$, which for $T \to \infty$ leads to the following result.

[5] In the classical under-parameterized (online streaming) case with white noise, the same theory can be used to establish convergence to the true parameter under the so-called Robbins–Monro conditions ($\sum_{i=1}^{\infty} \eta_i = \infty$, $\sum_{i=1}^{\infty} \eta_i^2 < \infty$) in a very direct and simple way (see [14]).

Proposition 45. *For any differentiable loss $l(\cdot)$, any initialization w_0, and any step size η, consider the SMD iterates given in Eq.* (8.15) *with respect to any strictly convex potential $\psi(\cdot)$. If the iterates converge to a solution $w_\infty \in \mathcal{W}$, then*

$$w_\infty = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0). \tag{8.26}$$

Remark. *In particular, for the initialization $w_0 = \arg\min_{w \in \mathbb{R}^m} \psi(w)$, if the iterates converge to a solution $w_\infty \in \mathcal{W}$, then*

$$w_\infty = \arg\min_{w \in \mathcal{W}} \psi(w). \tag{8.27}$$

An equivalent form of Proposition 45 has been shown recently in, e.g., [85].[6] Other implicit regularization results have been shown in [86, 194] for classification problems, which are not discussed here. Note that the result of [85] does not say anything about *whether the algorithm converges or not*. However, our fundamental identity of SMD (Lemma 42) allows us to also establish convergence to the regularized point, for some common cases, which will be shown next.

What Proposition 45 says is that depending on the choice of the potential function $\psi(\cdot)$, the optimization algorithm can perform an implicit regularization without any explicit regularization term. In other words, for any desired regularizer, if one chooses a potential function that approximates the regularizer, we can run the optimization without explicit regularization, and if it converges to a solution, the solution must be the one with the minimum potential.

In principle, one can choose the potential function in SMD for *any* desired convex regularization. For example, we can find the maximum entropy solution by taking the potential to be the negative entropy. Another illustrative example follows.

Example [Compressed Sensing]: In compressed sensing, one seeks the sparsest solution to an under-determined (over-parameterized) system of linear equations. The surrogate convex problem one solves is:

$$\begin{aligned} \min \quad & \|w\|_1 \\ \text{subject to} \quad & y_i = x_i^T w, \quad i = 1, \dots n \end{aligned} \tag{8.28}$$

One cannot choose $\psi(w) = \|w\|_1$, since it is neither differentiable nor strictly convex. However, $\psi(w) = \|w\|_{1+\epsilon}$, for any $\epsilon > 0$, can be used. Figure 4 shows a compressed

[6] To be precise, the authors in [85] assume convergence to a global minimizer of the loss function $L(w) = \sum_{i=1}^n l(y_i - x_i^T w)$, which, with their assumption of the loss function $l(\cdot)$ having a unique finite root, is equivalent to assuming convergence to a point $w_\infty \in \mathcal{W}$.

sensing example, with $n = 50$, $m = 100$, and sparsity $k = 10$. SMD was used with a step size of $\eta = 0.001$ and the potential function was $\psi(\cdot) = \|\cdot\|_{1.1}$. SMD converged to the true sparse solution after around 10,000 iterations. On this example, it was an order of magnitude faster than standard l_1 optimization.

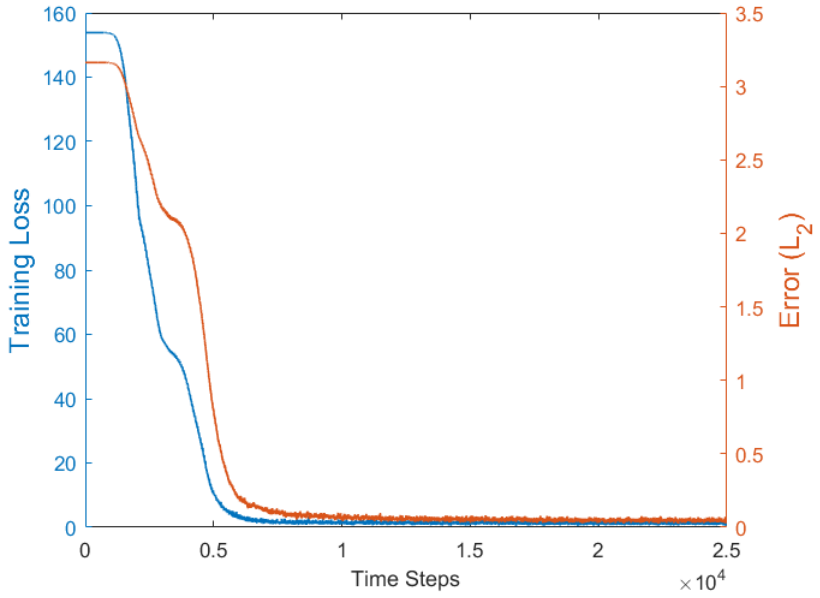


Figure 8.2: The training loss and actual error of stochastic mirror descent for compressed sensing. SMD recovers the actual sparse signal.

Next we establish *convergence to the regularized point* for the convex case.

Proposition 46. *Consider the following two cases.*

(i) $l(\cdot)$ *is differentiable and convex and has a unique root at 0,* $\psi(\cdot)$ *is strictly convex, and* $\eta > 0$ *is such that* $\psi - \eta L_i$ *is convex for all* i*.*

(ii) $l(\cdot)$ *is differentiable and quasi-convex,* $l'(\cdot)$ *is zero only at zero,* $\psi(\cdot)$ *is* α*-strongly convex, and* $0 < \eta \leq \min_i \frac{\alpha|y_i - x_i^T w_{i-1}|}{\|x_i\|^2 |l'(y_i - x_i^T w_{i-1})|}$.

If either (i) or (ii) holds, then for any w_0*, the SMD iterates given in Eq.* (8.15) *converge to*

$$w_\infty = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0). \tag{8.29}$$

The proof is provided in Appendix 8.C.

8.6 Concluding Remarks

We should remark that all the results stated throughout the chapter extend to the case of time-varying step size η_i, with minimal modification. In particular, it is easy to show that in this case, the identity (the counterpart of Eq. (8.19)) becomes

$$D_\psi(w, w_0) + \sum_{i=1}^{T} \eta_i l(v_i) = D_\psi(w, w_T) + \sum_{i=1}^{T} \left(E_i(w_i, w_{i-1}) + \eta_i D_{L_i}(w, w_{i-1}) \right), \tag{8.30}$$

where $E_i(w_i, w_{i-1}) = D_\psi(w_i, w_{i-1}) - \eta_i D_{L_i}(w_i, w_{i-1}) + \eta_i L_i(w_i)$. As a consequence, our main result will be the same as in Theorem 43, with the only difference being that the small-step-size condition in this case is the convexity of $\psi(w) - \eta_i L_i(w)$ for all i, and the SMD with time-varying step size will be the optimal solution to the following minimax problem

$$\min_{\{w_i\}} \max_{w, \{v_i\}} \frac{D_\psi(w, w_T) + \sum_{i=1}^{T} \eta_i D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \sum_{i=1}^{T} \eta_i l(v_i)}. \tag{8.31}$$

Similarly, the convergence and implicit regularization results can be proven under the same conditions (See Appendix 8.D for more details on the time-varying case).

This work opens up a variety of important directions for future work. Most of the analysis developed here is general, in terms of the *model*, the *loss function*, and the *potential function*. Therefore, it would be interesting to study the implications of this theory for specific classes of models (such as different neural networks), specific losses, and specific mirror maps (which induce different regularization biases).

8.A Proof of Lemma 41 (Fundamental Identity)

Proof. Let us start by expanding the Bregman divergence $D_\psi(w, w_i)$ based on its definition

$$D_\psi(w, w_i) = \psi(w) - \psi(w_i) - \nabla\psi(w_i)^T(w - w_i).$$

By plugging the SMD update rule $\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta\nabla L_i(w_{i-1})$ into this, we can write it as

$$D_\psi(w, w_i) = \psi(w) - \psi(w_i) - \nabla\psi(w_{i-1})^T(w - w_i) + \eta\nabla L_i(w_{i-1})^T(w - w_i). \quad (8.32)$$

Using the definition of Bregman divergence for (w, w_{i-1}) and (w_i, w_{i-1}), i.e., $D_\psi(w, w_{i-1}) = \psi(w) - \psi(w_{i-1}) - \nabla\psi(w_{i-1})^T(w - w_{i-1})$ and $D_\psi(w_i, w_{i-1}) = \psi(w_i) - \psi(w_{i-1}) - \nabla\psi(w_{i-1})^T(w_i - w_{i-1})$, we can express this as

$$\begin{aligned} D_\psi(w, w_i) &= D_\psi(w, w_{i-1}) + \psi(w_{i-1}) + \nabla\psi(w_{i-1})^T(w - w_{i-1}) - \psi(w_i) \\ &\qquad - \nabla\psi(w_{i-1})^T(w - w_i) + \eta\nabla L_i(w_{i-1})^T(w - w_i) \quad (8.33) \\ &= D_\psi(w, w_{i-1}) + \psi(w_{i-1}) - \psi(w_i) + \nabla\psi(w_{i-1})^T(w_i - w_{i-1}) \\ &\qquad + \eta\nabla L_i(w_{i-1})^T(w - w_i) \quad (8.34) \\ &= D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta\nabla L_i(w_{i-1})^T(w - w_i). \quad (8.35) \end{aligned}$$

Expanding the last term using $w - w_i = (w - w_{i-1}) - (w_i - w_{i-1})$, and following the definition of $D_{L_i}(.,.)$ from (8.16) for (w, w_{i-1}) and (w_i, w_{i-1}), we have

$$\begin{aligned} D_\psi(w, w_i) &= D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta\nabla L_i(w_{i-1})^T(w - w_{i-1}) \\ &\qquad - \eta\nabla L_i(w_{i-1})^T(w_i - w_{i-1}) \quad (8.36) \\ &= D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta\left(L_i(w) - L_i(w_{i-1}) - D_{L_i}(w, w_{i-1})\right) \\ &\qquad - \eta\left(L_i(w_i) - L_i(w_{i-1}) - D_{L_i}(w_i, w_{i-1})\right) \quad (8.37) \\ &= D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta\left(L_i(w) - D_{L_i}(w, w_{i-1})\right) \\ &\qquad - \eta\left(L_i(w_i) - D_{L_i}(w_i, w_{i-1})\right). \quad (8.38) \end{aligned}$$

Defining $E_i(w_i, w_{i-1}) := D_\psi(w_i, w_{i-1}) - \eta D_{L_i}(w_i, w_{i-1}) + \eta L_i(w_i)$, we can write the above equality as

$$D_\psi(w, w_i) = D_\psi(w, w_{i-1}) - E_i(w_i, w_{i-1}) + \eta\left(L_i(w) - D_{L_i}(w, w_{i-1})\right). \quad (8.39)$$

Notice that for any model class with additive noise, and any loss function L_i that depends only on the residual (i.e., the difference between the prediction and the true label), the term $L_i(w)$ depends only on the noise term, for any "true" parameter w. In other words, for all w that satisfy $y_i = f(x_i, w) + v_i$, we have $L_i(w) = l(y_i - f(x_i, w)) = l(y_i - (y_i - v_i)) = l(v_i)$. Finally, reordering the terms leads to

$$D_\psi(w, w_i) + \eta D_{L_i}(w, w_{i-1}) + E_i(w_i, w_{i-1}) = D_\psi(w, w_{i-1}) + \eta l(v_i), \tag{8.40}$$

which concludes the proof. □

8.B Proof of Theorem 43 (Minimax Optimality)

Proof. We prove the theorem in two parts. First, we show that the value of the minimax is at least 1. Then we prove that the value is at most 1, and is achieved by stochastic mirror descent for small enough step size.

1. Consider the maximization problem

$$\max_{w,\{v_i\}} \frac{D_\psi(w, w_T) + \eta \sum_{i=1}^T D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \eta \sum_{i=1}^T l(v_i)}.$$

Clearly, the optimal solution(s) and the optimal value of this problem can, and will, be a function of $\{w_i\}$. Similarly, we can also choose feasible points that depend on $\{w_i\}$. Any choice of a feasible point $(\hat{w}, \{\hat{v}_i\})$ gives a lower bound on the value of the problem. Before choosing a feasible point, let us first expand the $D_{L_i}(w, w_{i-1})$ term in the numerator, according to its definition.

$$D_{L_i}(w, w_{i-1}) = l(v_i) - l(y_i - f_i(w_{i-1})) + l'(y_i - f_i(w_{i-1}))\nabla f(w_{i-1})^T(w - w_{i-1}), \tag{8.41}$$

where we have used the fact that $l(y_i - f_i(w)) = l(v_i)$ for all consistent w, in the first term.

Now, we choose a feasible point as follows

$$\hat{v}_i = f_i(w_{i-1}) - f_i(\hat{w}), \tag{8.42}$$

where $\hat{w}$ is the choice of w, as will be described soon. The reason for choosing this value for the noise is that it "fools" the estimator by making its loss on the corresponding data point zero. In other words, for this choice, we have

$$\begin{aligned} D_{L_i}(w, w_{i-1}) &= l(\hat{v}_i) - l(0) + l'(0)\nabla f(w_{i-1})^T(\hat{w} - w_{i-1}) \\ &= l(\hat{v}_i) \end{aligned}$$

because $l(0) = l'(0) = 0$. It should be clear at this point that this choice makes the second terms in the numerator and the denominator equal, independent of the choice of $\hat{w}$. What remains to do, in order to show the 1 lower-bound, is to take care of the other two terms, i.e., $D_\psi(w, w_T)$ and $D_\psi(w, w_0)$. As we would like to make the ratio equal to one, we would like to have $D_\psi(w, w_T) = D_\psi(w, w_0)$, which is equivalent to having

$$\psi(w) - \psi(w_T) - \nabla\psi(w_T)^T(w - w_T) = \psi(w) - \psi(w_0) - \nabla\psi(w_0)^T(w - w_0)$$

which is, in turn, equivalent to

$$(\nabla\psi(w_T) - \nabla\psi(w_0))^T w = -\psi(w_T) + \psi(w_0) + \nabla\psi(w_T)^T w_T - \nabla\psi(w_0)^T w_0. \tag{8.43}$$

Since $\nabla\psi$ is an invertible function, $\nabla\psi(w_T) - \nabla\psi(w_0) \neq 0$, if $w_T \neq w_0$. Therefore, the above equation has a solution for w, if $w_T \neq w_0$. As a result, choosing $\hat{w}$ to be a solution to (8.43) makes $D_\psi(\hat{w}, w_T) = D_\psi(\hat{w}, w_0)$, if $w_T \neq w_0$. For the case when $w_T = w_0$, it is trivial that $D_\psi(\hat{w}, w_T) = D_\psi(\hat{w}, w_0)$ for any choice of $\hat{w}$. In this case, we only need to choose $\hat{w}$ to be different from w_0, to avoid making the ratio $\frac{0}{0}$. Hence, we have the following choice

$$\hat{w} = \begin{cases} \text{a solution of (8.43)} & \text{for } w_T \neq w_0 \\ w_0 + \delta w \ \text{ for some } \delta w \neq 0 & \text{for } w_T = w_0 \end{cases} \tag{8.44}$$

Choosing the feasible point $\hat{w}, \{v_i\}$ according to (8.44) and (8.42) leads to

$$\max_{w,\{v_i\}} \frac{D_\psi(w, w_T) + \eta \sum_{i=1}^T D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \eta \sum_{i=1}^T l(v_i)} \geq \frac{D_\psi(\hat{w}, w_T) + \eta \sum_{i=1}^T l(f_i(w_{i-1}) - f_i(\hat{w}))}{D_\psi(\hat{w}, w_0) + \eta \sum_{i=1}^T l(f_i(w_{i-1}) - f_i(\hat{w}))}. \tag{8.45}$$

Taking the minimum of both sides with respect to $\{w_i\}$, we have

$$\min_{\{w_i\}} \max_{w,\{v_i\}} \frac{D_\psi(w, w_T) + \eta \sum_{i=1}^T D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \eta \sum_{i=1}^T l(v_i)} \geq \min_{\{w_i\}} \frac{D_\psi(\hat{w}, w_T) + \eta \sum_{i=1}^T l(f_i(w_{i-1}) - f_i(\hat{w}))}{D_\psi(\hat{w}, w_0) + \eta \sum_{i=1}^T l(f_i(w_{i-1}) - f_i(\hat{w}))} = 1. \tag{8.46}$$

The equality to 1 comes from the fact the that the optimal solution of the minimization either has $w_T^* = w_0$ or $w_T^* \neq w_0$, and in both cases the ratio is equal to 1.

2. Now we prove that, under the small step size condition (convexity of $\psi(w) - \eta L_i(w)$ for all i), SMD makes the minimax value at most 1, which means that it is indeed an optimal solution. Recall from Lemma 42 that

$$D_\psi(w, w_0) + \eta \sum_{i=1}^T l(v_i) = D_\psi(w, w_T) + \sum_{i=1}^T E_i(w_i, w_{i-1}) + \eta \sum_{i=1}^T D_{L_i}(w, w_{i-1}),$$

where

$$E_i(w_i, w_{i-1}) = D_\psi(w_i, w_{i-1}) - \eta D_{L_i}(w_i, w_{i-1}) + \eta L_i(w_i).$$

It is easy to check that when $\psi(w) - \eta L_i(w)$ is convex, $D_\psi(w_i, w_{i-1}) - \eta D_{L_i}(w_i, w_{i-1})$ is in fact a Bregman divergence (i.e., the Bregman divergence with respect to the potential $\psi(w) - \eta L_i(w)$), and therefore it is nonnegative for any w_i and w_{i-1}. Furthermore, we know that the loss $L_i(w_i)$ is also nonnegative for all w_i. It follows that $E_i(w_i, w_{i-1})$ is nonnegative for all values of w_i, w_{i-1} and i. As a result, we have the following bound.

$$D_\psi(w, w_0) + \eta \sum_{i=1}^{T} l(v_i) \geq D_\psi(w, w_T) + \eta \sum_{i=1}^{T} D_{L_i}(w, w_{i-1}). \tag{8.47}$$

Since the Bregman divergence $D_\psi(w, w_0)$ and the loss $l(v_i)$ are nonnegative, the left-hand side expression is nonnegative, and it follows that

$$\frac{D_\psi(w, w_T) + \eta \sum_{i=1}^{T} D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \eta \sum_{i=1}^{T} l(v_i)} \leq 1. \tag{8.48}$$

In fact, this means that, independent of the choice of the maximizer (i.e., for all $\{v_i\}$ and w), as long as the step size condition is met, SMD makes the ratio less than or equal to 1.

Combining the results of 1 and 2 above concludes the proof. □

8.B.1 Proof of Theorem 40

Proof. This result is a special case of Theorem 43, which was proven above. In this case, $\psi(w) = \frac{1}{2}\|w\|^2$, $f(x_i, w) = x_i^T w$, and $l(z) = \frac{1}{2}z^2$. Therefore, $D_\psi(w, w_T) = \frac{1}{2}\|w - w_T\|^2$, $D_\psi(w, w_0) = \frac{1}{2}\|w - w_0\|^2$, $D_{L_i}(w, w_{i-1}) = \frac{1}{2}(x_i^T w - x_i^T w_{i-1})^2$, and $l(v_i) = \frac{1}{2}v_i^2$, which leads to the result. □

8.C Proof of Proposition 46 (Convergence)

Proof. To prove convergence, we appeal again to Equation (8.22), i.e.,

$$D_\psi(w, w_0) = D_\psi(w, w_T) + \sum_{i=1}^{T} \left(E_i(w_i, w_{i-1}) + \eta D_{L_i}(w, w_{i-1}) \right), \tag{8.49}$$

for all $w \in \mathcal{W}$. We prove the two cases separately.

1. The proof of case (i) is straightforward. When $l(\cdot)$ is differentiable and convex, L_i is also convex, and therefore $D_{L_i}(w, w_{i-1})$ is nonnegative. Moreover, when $\psi - \eta L_i$ is convex, $E_i(w_i, w_{i-1})$ is also nonnegative. Therefore, the entire summand in Eq. (8.49) is nonnegative and thus has to go to zero for $i \to \infty$. That is because as $T \to \infty$, the sum should remain bounded, i.e., $\sum_{i=1}^{\infty} \left(E_i(w_i, w_{i-1}) + \eta D_{L_i}(w, w_{i-1}) \right) \leq D_\psi(w, w_0)$. As a result of the non-negativity of both terms in the sum, we have both $E_i(w_i, w_{i-1}) \to 0$ and $D_{L_i}(w, w_{i-1}) \to 0$ as $i \to \infty$, the latter of which implies $L_i(w_{i-1}) \to 0$. This implies that the updates in (8.15) vanish and we get convergence, i.e., $w_i \to w_\infty$. Further, again because $L_i(w_{i-1}) \to 0$, and 0 is the unique root of $l(\cdot)$, all the data point are being fit, which means $w_\infty \in \mathcal{W}$.

2. To prove case (ii), note that we have

$$\begin{aligned} D_{L_i}(w, w_{i-1}) &= L_i(w) - L_i(w_{i-1}) - \nabla L_i(w_{i-1})^T (w - w_{i-1}) && (8.50)\\ &= 0 - l(y_i - x_i^T w_{i-1}) + l'(y_i - x_i^T w_{i-1}) x_i^T (w - w_{i-1}) && (8.51)\\ &= -l(y_i - x_i^T w_{i-1}) + l'(y_i - x_i^T w_{i-1})(y_i - x_i^T w_{i-1}), && (8.52) \end{aligned}$$

and

$$\begin{aligned} E_i(w_i, w_{i-1}) &= D_\psi(w_i, w_{i-1}) - \eta D_{L_i}(w_i, w_{i-1}) + \eta L_i(w_i) && (8.53)\\ &= D_\psi(w_i, w_{i-1}) + \eta \left(L_i(w_{i-1}) + \nabla L_i(w_{i-1})^T (w_i - w_{i-1}) \right) && (8.54)\\ &= D_\psi(w_i, w_{i-1}) + \eta \left(l(y_i - x_i^T w_{i-1}) - l'(y_i - x_i^T w_{i-1}) x_i^T (w_i - w_{i-1}) \right). && (8.55) \end{aligned}$$

It follows from (8.52) and (8.55) that the summand in Equation (8.49) is

$$E_i(w_i, w_{i-1}) + \eta D_{L_i}(w, w_{i-1}) = D_\psi(w_i, w_{i-1}) + \eta l'(y_i - x_i^T w_{i-1})(y_i - x_i^T w_i). \tag{8.56}$$

The first term is a Bregman divergence and is therefore nonnegative. In order to establish convergence, one needs to argue that the second term is nonnegative as well, so that the summand goes to zero as $i \to \infty$. Since $l(\cdot)$ is increasing for positive values and decreasing for negative values, it is enough to show that $y_i - x_i^T w_{i-1}$ and $y_i - x_i^T w_i$ have the same sign, in order to establish nonnegativity. It is not hard to see that if the distance between the two points is less than or equal to the distance of $y_i - x_i^T w_i$ from the origin, then the signs are the same. In other words, if $|(y_i - x_i^T w_i) - (y_i - x_i^T w_{i-1})| = |x_i^T (w_i - w_{i-1})| \leq |y_i - x_i^T w_{i-1}|$, then the sign are the same.

Note that by the definition of α-strong convexity of $\psi(\cdot)$, we have

$$(\nabla\psi(w_i) - \nabla\psi(w_{i-1}))^T (w_i - w_{i-1}) \geq \alpha \|w_i - w_{i-1}\|^2, \tag{8.57}$$

which implies

$$-\eta \nabla L_i(w_{i-1})^T (w_i - w_{i-1}) \geq \alpha \|w_i - w_{i-1}\|^2, \tag{8.58}$$

by substituting from the SMD update rule. Upper-bounding the left-hand side by $\eta \|\nabla L_i(w_{i-1})\| \|(w_i - w_{i-1})\|$ implies

$$\eta \|\nabla L_i(w_{i-1})\| \geq \alpha \|w_i - w_{i-1}\|. \tag{8.59}$$

This implies that we have the following bound

$$|x_i^T (w_i - w_{i-1})| \leq \|x_i\| \|w_i - w_{i-1}\| \leq \frac{\eta \|x_i\| \|\nabla L_i(w_{i-1})\|}{\alpha}. \tag{8.60}$$

It follows that if $\eta \leq \frac{\alpha |y_i - x_i^T w_{i-1}|}{\|x_i\| \|\nabla L_i(w_{i-1})\|}$, for all i, then the signs are the same, and the summand in Eq.(8.49) is indeed nonnegative. This condition can be equivalently expressed as $\eta \leq \frac{\alpha |y_i - x_i^T w_{i-1}|}{\|x_i\|^2 |l'(y_i - x_i^T w_{i-1})|}$ for all i, or $\eta \leq \min_i \frac{\alpha |y_i - x_i^T w_{i-1}|}{\|x_i\|^2 |l'(y_i - x_i^T w_{i-1})|}$, which is the condition in the statement of the proposition.

Now that we have argued that the summand is nonnegative, the convergence to $w_\infty \in \mathcal{W}$ is immediate. The reason is that both $D_\psi(w_i, w_{i-1}) \to 0$ and $l'(y_i - x_i^T w_{i-1})(y_i - x_i^T w_i) \to 0$, as $i \to \infty$. The first one implies convergence to a point w_∞. The second one implies that either $y_i - x_i^T w_{i-1} = 0$ or $y_i - x_i^T w_i = 0$, which, in turn, implies $w_\infty \in \mathcal{W}$.

□

8.D Time-Varying Step-Size

The update rule for the stochastic mirror descent with time-varying step size is as follows.

$$w_i = \arg\min_{w} \ \eta_i w^T \nabla L_i(w_{i-1}) + D_\psi(w, w_{i-1}), \tag{8.61}$$

which can be equivalently expressed as $\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta_i \nabla L_i(w_{i-1})$, for all i. The main results in this case are as follows.

Lemma 47. *For any (nonlinear) model $f(\cdot,\cdot)$, any differentiable loss $l(\cdot)$, any parameter w and noise values $\{v_i\}$ that satisfy $y_i = f(x_i, w) + v_i$ for $i = 1, \ldots, n$, any initialization w_0, any step size sequence $\{\eta_i\}$, and any number of steps $T \geq 1$, the following relation holds for the SMD iterates $\{w_i\}$ given in Eq.* (8.61)

$$D_\psi(w, w_0) + \sum_{i=1}^{T} \eta_i l(v_i) = D_\psi(w, w_T) + \sum_{i=1}^{T} \left(E_i(w_i, w_{i-1}) + \eta_i D_{L_i}(w, w_{i-1}) \right), \tag{8.62}$$

Proof. The proof is straightforward by summing the following equation for all $i = 1, \ldots, T$

$$D_\psi(w, w_{i-1}) + \eta_i l(v_i) = D_\psi(w, w_i) + E_i(w_i, w_{i-1}) + \eta_i D_{L_i}(w, w_{i-1}), \tag{8.63}$$

which can be easily shown in the same way as in the proof of Lemma 41 in Appendix 8.A. □

Theorem 48. *Consider any general model $f(\cdot,\cdot)$, and any differentiable loss function $l(\cdot)$ with property $l(0) = l'(0) = 0$. For sufficiently small step size, i.e., for any sequence $\{\eta_i\}$ for which $\psi(w) - \eta_i L_i(w)$ is convex for all i, the SMD iterates $\{w_i\}$ given by Eq.* (8.61) *are the optimal solution to the following minimization problem*

$$\min_{\{w_i\}} \max_{w, \{v_i\}} \frac{D_\psi(w, w_T) + \sum_{i=1}^{T} \eta_i D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \sum_{i=1}^{T} \eta_i l(v_i)}. \tag{8.64}$$

Furthermore, the optimal value (achieved by SMD) is 1.

Proof. The proof is similar to that of Theorem 43, as presented in Appendix 8.B. The argument for the upper-bound of 1 is exactly the same. For the second part of the proof, we use the previous Lemma. It follows from the convexity of $\psi(w) - \eta_i L_i(w)$ that $E_i(w_i, w_{i-1}) \geq 0$, and as a result we have

$$\frac{D_\psi(w, w_T) + \sum_{i=1}^{T} \eta_i D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \sum_{i=1}^{T} \eta_i l(v_i)} \leq 1 \tag{8.65}$$

for SMD updates, which concludes the proof. □

The convergence and implicit regularization results hold similarly, and can be formally stated as follows.

Proposition 49. *Consider the following two cases.*

(i) $l(\cdot)$ *is differentiable and convex and has a unique root at 0,* $\psi(\cdot)$ *is strictly convex, and the positive sequence* $\{\eta_i\}$ *is such that* $\psi - \eta_i L_i$ *is convex for all* i*.*

(ii) $l(\cdot)$ *is differentiable and quasi-convex and has zero derivative only at 0,* $\psi(\cdot)$ *is* α*-strongly convex, and* $0 < \eta_i \leq \frac{\alpha|y_i - x_i^T w_{i-1}|}{\|x_i\|^2 |l'(y_i - x_i^T w_{i-1})|}$ *for all* i*.*

If either (i) or (ii) holds, then for any initialization w_0*, the SMD iterates given in Eq.* (8.61) *converge to*

$$w_\infty = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0). \tag{8.66}$$

Proof. The proof is similar to that of Proposition 46, as provided in Appendix 8.C. □

Chapter 9

SMD ON OVERPARAMETERIZED NONLINEAR MODELS

[1] Navid Azizan et al. "A Study of Generalization of Stochastic Mirror Descent Algorithms on Overparameterized Nonlinear Models". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 3132–3136. DOI: 10.1109/ICASSP40776.2020.9053864.

[2] Navid Azizan et al. "Stochastic Mirror Descent on Overparameterized Nonlinear Models: Convergence, Implicit Regularization, and Generalization". In: *2019 International Conference on Machine Learning (ICML) Generalization Workshop*. 2019.

Most modern learning problems are highly overparameterized, i.e., the model has many more parameters than the number of training data points, and the training loss has infinitely many global minima. Therefore, it is important to understand which interpolating solutions we converge to, how they depend on the initialization and learning algorithm, and whether they yield different generalization errors. In this chapter, we study these questions for the family of stochastic mirror descent (SMD) algorithms, of which stochastic gradient descent (SGD) is a special case. As we saw in the previous chapter, for overparameterized *linear* models, SMD converges to the closest global minimum to the initialization point, where closeness is in terms of the Bregman divergence corresponding to the potential function of the mirror descent. For initialization points around "zero" (i.e., the minimizer of the potential), this means convergence to the minimum-potential interpolating solution, a phenomenon referred to as implicit regularization. Our contributions in this chapter are both theoretical and experimental. On the theory side, we show that for overparameterized *nonlinear* models, if the model is sufficiently overparameterized so that a random initialization is w.h.p. close to the manifold of global minima, SMD with a (sufficiently small) *fixed* step size converges to a global minimum that is approximately the closest one in Bregman divergence, thus attaining *approximate implicit regularization*. On the experimental side, our extensive experiments on the MNIST and CIFAR-10 datasets consistently confirm that this phenomenon occurs in practical scenarios. They further indicate a clear difference in the generalization performances of different SMD algorithms: experiments on the CIFAR-10 dataset with different regularizers, ℓ_1 to encourage sparsity, ℓ_2 (SGD) to encourage small

Euclidean norm, and ℓ_{10} to discourage large components, consistently show that ℓ_{10}-SMD has better generalization performance than SGD, which in turn generalizes better than ℓ_1-SMD.

9.1 Introduction

Deep learning has demonstrably enjoyed a great deal of success in a wide variety of tasks [9, 80, 118, 143, 189, 213, 119]. Despite its tremendous success, the reasons behind the good performance of these methods on unseen data is not fully understood (and, arguably, remains somewhat of a mystery). While the special deep architecture of these models seems to be important to the success of deep learning, the architecture is only part of the story, and it has been now widely recognized that the optimization algorithms used to train these models, typically stochastic gradient descent (SGD) and its variants, play a key role in learning parameters that generalize well.

Since these deep models are *highly overparameterized*, they have a lot of capacity, and can fit to virtually any (even random) set of data points [224]. In other words, these highly overparameterized models can "interpolate" the training data, so much so that this regime has been called the "interpolating regime" [138]. In fact, on a given dataset, the loss function typically has (infinitely) many *global minima*, which, however, can have drastically different generalization properties (many of them perform poorly on the test set). Which minimum among all the possible minima we converge to in practice is determined by the initialization and the optimization algorithm that we use for training the model.

Since the loss functions of deep neural networks are non-convex—sometimes even non-smooth—in theory, one may expect the optimization algorithms to get stuck in local minima or saddle points. In practice, however, such simple stochastic descent algorithms almost always reach *zero training error*, i.e., a *global minimum* of the training loss [224, 122]. More remarkably, even in the absence of any explicit regularization, dropout, or early stopping [224], the global minima obtained by these algorithms seem to generalize quite well (contrary to some other "bad" global minima). It has been also observed that even among different optimization algorithms, i.e., SGD and its variants, there is a discrepancy in the solutions achieved by different algorithms and how they generalize [211].

In this chapter, we propose training deep neural networks with the family of stochastic mirror descent (SMD) algorithms, which is a generalization of the popular SGD. For

any choice of potential function, there is a corresponding mirror descent algorithm. We train a standard ResNet-18 architecture on CIFAR-10 using mirror descents with the following four different potential functions: ℓ_1 norm, ℓ_2 norm (SGD), ℓ_3 norm, and ℓ_{10} norm. In all the cases, we train the network for a sufficiently large number of steps, with a sufficiently small step size, until we converge to an interpolating solution (global minima). Comparisons between the histograms of these different global minima show that they are vastly different. In particular, the solution obtained by ℓ_1-SMD is very sparse, and on the contrary, the solution obtained by the ℓ_{10} does not have any zero components. More importantly, there is a clear gap in the generalization performance of these algorithms. In fact, surprisingly and somewhat counterintuitively, the solution obtained by the ℓ_{10}-SMD, which uses the entire overparameterization in the network, consistently outperforms SGD, which in turn performs better than the SMD with ℓ_1 norm, i.e., the sparser one. Therefore, it is important to ask: *Which global minima do these algorithms converge to, and what properties do they have?*

On the theory side, we show that, for overparameterized nonlinear models, if the model is sufficiently overparameterized so that the random initialization point is w.h.p. close to the manifold of interpolating solutions (something that is occasionally referred to as "the blessing of dimensionality"), then the SMD algorithm for any particular potential function converges to a global minimum that is approximately *the closest one to the initialization, in Bregman divergence corresponding to the potential.* For the special case of SGD, this means that it converges to a global minimum which is approximately the closest one to the initialization in the usual Euclidean sense.

We perform extensive systematic experiments with various initial points and various mirror descent algorithms for the MNIST and CIFAR-10 datasets using standard off-the-shelf deep neural network architectures for these datasets with standard random initialization, and we measure all the resulting pairwise Bregman divergences. We found that every single result is exactly consistent with the above theory. Indeed, in all our experiments, *the global minimum achieved by any particular mirror descent algorithm is the closest, among all other global minima obtained by other mirrors and other initializations, to its initialization in the corresponding Bregman divergence.* In particular, the global minimum obtained by SGD from any particular initialization is closest to the initialization in Euclidean sense, both among the global minima obtained by different mirrors and among the global minima obtained by different

initializations.

This result, proven theoretically and backed up by extensive experiments, further implies that, even in the absence of any explicit regularization, these algorithms perform an *implicit regularization*. In particular, it implies that, when initialized around zero, SGD acts as an approximate ℓ_2-norm regularizer on the weights. Similarly, by choosing other mirrors, one can obtain any desired form of implicit regularization (such as ℓ_1 or ℓ_∞), which is consistent with the observations about the histograms.

9.2 Background

Let us denote the training dataset by $\{(x_i, y_i) : i = 1, \dots, n\}$, where $x_i \in \mathbb{R}^d$ are the inputs, and $y_i \in \mathbb{R}$ are the labels. The model (which can be, e.g., linear, a deep neural network, etc.) is defined by the general function $f(x_i, w) = f_i(w)$ with some parameter vector $w \in \mathbb{R}^m$. Since typical deep models have a lot of capacity and are highly overparameterized, we are particularly interested in the overparameterized (or so-called interpolating) regime, where $m > n$ (often $m \gg n$). In this case, there are many parameter vectors w that are consistent with the training data points. We denote the set of these parameter vectors by

$$\mathcal{W} = \{w \in \mathbb{R}^m \mid f(x_i, w) = y_i, i = 1, \dots, n\}. \tag{9.1}$$

This is a high-dimensional set (e.g., a $(m - n)$-dimensional manifold) in $\mathbb{R}^m$ and depends only on the training data $\{(x_i, y_i) : i = 1, \dots, n\}$ and the model $f(\cdot, \cdot)$.

The total loss on the training set (empirical risk) can be expressed as $L(w) = \sum_{i=1}^n L_i(w)$, where $L_i(\cdot) = \ell(y_i, f(x_i, w))$ is the loss on the individual data point i, and $\ell(\cdot, \cdot)$ is a differentiable non-negative function, with the property that $\ell(y_i, f(x_i, w)) = 0$ iff $y_i = f(x_i, w)$. Often $\ell(y_i, f(x_i, w)) = \ell(y_i - f(x_i, w))$, with $\ell(\cdot)$ convex and having a global minimum at zero (such as square loss, Huber loss, etc.). In this case, $L(w) = \sum_{i=1}^n \ell(y_i - f(x_i, w))$. For example, the conventional gradient descent (GD) algorithm can be used as an attempt to minimize $L(\cdot)$ over w.

An important generalization of GD is the *mirror descent* (MD) algorithm, first introduced by Nemirovski and Yudin [149] and widely used since then [33, 55, 227], can be described as follows. Consider a strictly convex differentiable function $\psi(\cdot)$, called the *potential function*. Then MD is given by the following recursion

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta\nabla L(w_{i-1}), \quad w_0 \tag{9.2}$$

where $\eta > 0$ is known as the step size or learning rate. Note that, due to the strict convexity of $\psi(\cdot)$, the gradient $\nabla\psi(\cdot)$ defines an invertible map so that the recursion in (9.2) yields a unique w_i at each iteration, i.e., $w_i = \nabla\psi^{-1}\left(\nabla\psi(w_{i-1}) - \eta\nabla L(w_{i-1})\right)$. Compared to classical GD, rather than update the weight vector along the direction of the negative gradient, the update is done in the "mirrored" domain determined by the invertible transformation $\nabla\psi(\cdot)$. Mirror descent was originally conceived to exploit the geometrical structure of the problem by choosing an appropriate potential. Note that MD reduces to GD when $\psi(w) = \frac{1}{2}\|w\|^2$, since the gradient is simply the identity map. Other examples include the exponentiated gradient descent (also known as the exponential weights) and the p-norms algorithm [82, 78].

When n is large, computation of the entire gradient may be cumbersome. Alternatively, in online scenarios, the entire loss function $L(\cdot)$ may not be available, and only the local loss functions may be provided at each iteration. In such settings, a stochastic version of MD has been introduced, aptly called *stochastic mirror descent* (SMD), which can be considered the straightforward generalization of stochastic gradient descent (SGD):

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta\nabla L_i(w_{i-1}), \quad w_0 \tag{9.3}$$

In the offline setting, the various instantaneous loss functions $L_i(\cdot)$ can either be drawn at random or cycled through periodically.

Alternatively, the update rule (9.3) can be expressed as

$$w_i = \arg\min_w \ \eta w^T \nabla L_i(w_{i-1}) + D_\psi(w, w_{i-1}), \tag{9.4}$$

where

$$D_\psi(w, w_{i-1}) := \psi(w) - \psi(w_{i-1}) - \nabla\psi(w_{i-1})^T(w - w_{i-1}) \tag{9.5}$$

is the Bregman divergence with respect to the potential function $\psi(\cdot)$. Note that $D_\psi(\cdot,\cdot)$ is non-negative, convex in its first argument, and that, due to strict convexity, $D_\psi(w, w') = 0$ iff $w = w'$.

9.3 Training Deep Neural Networks with SMD

As mentioned earlier, the heavy overparameterization in typical deep neural networks means that the loss function for such architectures typically has infinitely many global minima, and these different minima can have very different properties and generalization performances. Motivated by this fact, we propose training deep neural networks with other members of the family of stochastic mirror descent, to see if they lead to different global minima.

We take the popular CIFAR-10 dataset and the standard ResNet-18 architecture for this dataset. We initialize the network with small random weights and train it with mirror descents with the following 4 different potential functions: ℓ_1 norm, ℓ_2 norm (SGD), ℓ_3 norm, and ℓ_{10} norm. In all the cases, we choose the step size to be sufficiently small, and we train for a sufficiently large number of steps, so that we converge to an interpolating solution (global minimum).

We compare the generalization performance of these different solutions on the test set. Fig. 9.1 shows the test accuracies of different algorithms with eight random initializations around zero. There is a clear gap in the generalization performance of these algorithms, and SMD with ℓ_{10}-norm consistently performs better than SGD, which in turn performs better than the SMD with ℓ_1-norm. In fact, perhaps surprisingly, by virtue of changing the optimizer from SGD to ℓ_{10}-SMD, without any additional tricks, we outperform the state of the art for ResNet-18 on CIFAR-10. This is particularly remarkable given that this very architecture had been designed with training with SGD in mind.

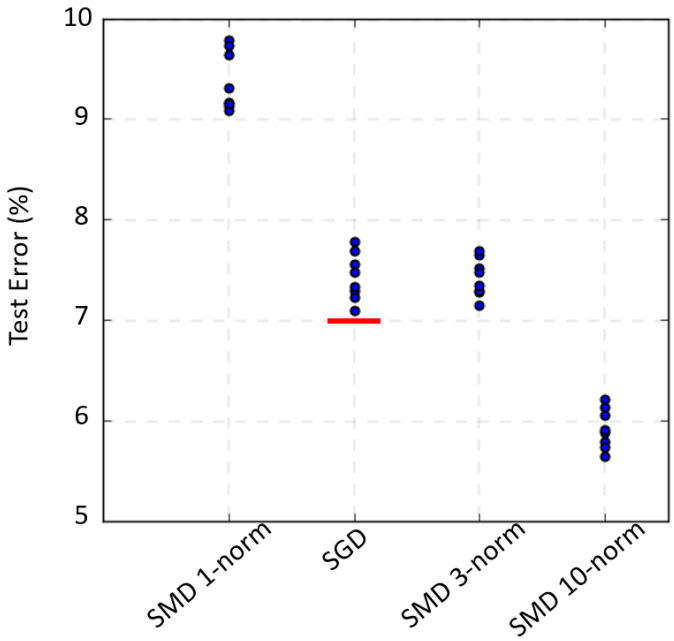


Figure 9.1: Generalization performance of different SMD algorithms on the CIFAR-10 dataset using ResNet-18. ℓ_{10} performs consistently better, while ℓ_1 performs consistently worse. The red line shows the state of the art on ResNet-18 for CIFAR-10 (93.02%)[135].

One may be curious to see how the weights obtained by these different mirrors

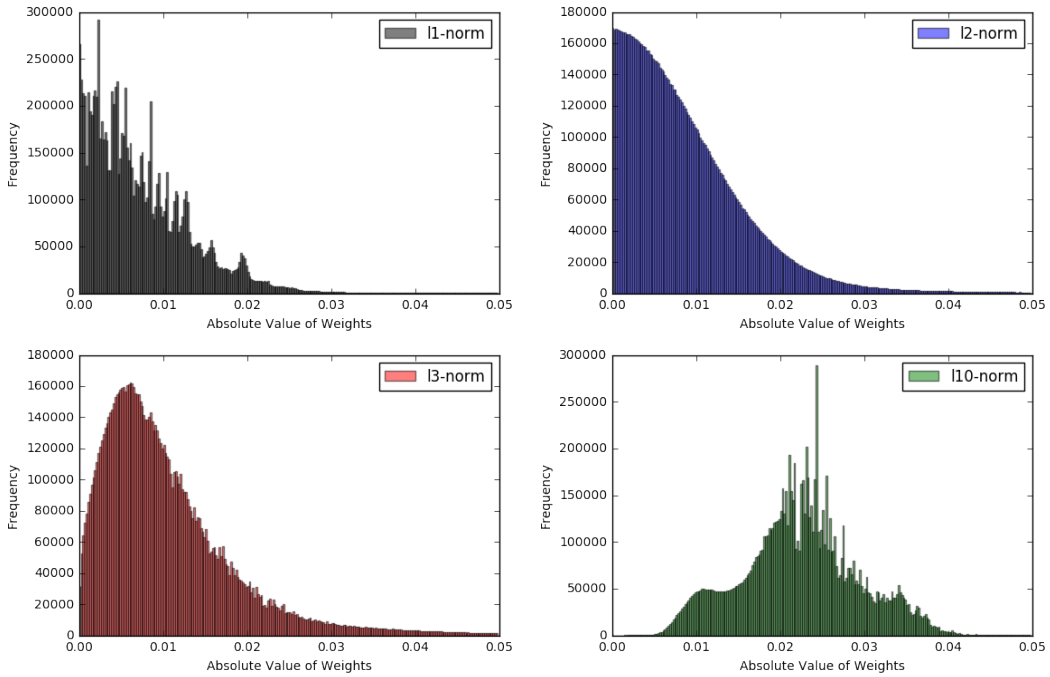


Figure 9.2: Histogram of the absolute value of the final weights in the network for different SMD algorithm with different potentials. Note that each of the four histograms corresponds to an 11×10^6-dimensional weight vector that perfectly interpolates the data. Even though the weights remain quite small, the histograms are drastically different. ℓ_1-SMD induces sparsity on the weights. SGD appears to lead to a Gaussian distribution on the weights. ℓ_3-SMD starts to reduce the sparsity, and ℓ_{10} shifts the distribution of the weights significantly, so much so that almost all the weights are non-zero.

look. Fig. 9.2 shows the histogram of the absolute value of the weights for these four different SMDs, initialized by the exact *same* set of weights. The histograms of the final weights look substantially different and, since they all started from the same initial weights and they all interpolate the same data set, this difference is fully attributable to the different mirrors used. The histogram of the ℓ_1-SMD has more weights at and around zero, i.e., it is very sparse. The histogram of the ℓ_2-SMD (SGD) looks almost perfectly Gaussian. The one corresponding to ℓ_3 has somewhat shifted to the right, and the ℓ_{10} has has completely moved away from zero, i.e., all the weights in the ℓ_{10} solution are non-zero. The fact that the ℓ_{10} solution, which uses the entire overparameterization available in the network, generalizes better than the sparser ones is very surprising.

9.4 Theoretical Results

In this section, we provide our main theoretical results. In particular, we show that for highly overparameterized models: (1) SMD converges to a global minimum and (2) the global minimum obtained by SMD is approximately the closest one to the initialization in Bregman divergence corresponding to the potential.

9.4.1 Warm-up: Overparameterized Linear Models

Overparameterized (or underdetermined) linear models have been recently studied in many papers due to their simplicity and the fact that there are interesting insights that one can obtain from them. In this case, the model is $f(x_i, w) = x_i^T w$, the set of global minima is $\mathcal{W} = \left\{w \mid y_i = x_i^T w,\ i = 1, \dots, n\right\}$, and the loss is $L_i(w) = \ell(y_i - x_i^T w)$. The following result characterizes the solution that SMD converges to [18, 85].

Proposition 50. *Consider a linear overparameterized model. For sufficiently small step size, i.e., for any $\eta > 0$ for which $\psi(\cdot) - \eta L_i(\cdot)$ is convex, and for any initialization w_0, the SMD iterates converge to*

$$w_\infty = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0).$$

Note that the step size condition, i.e., the convexity of $\psi(\cdot) - \eta L_i(\cdot)$, depends on both the loss and the potential function. For the case of SGD, $\psi(w) = \frac{1}{2}\|w\|^2$, and $\ell(y_i - x_i^T w) = \frac{1}{2}(y_i - x_i^T w)^2$, so the condition reduces to the well-known $\eta \leq \frac{1}{\|x_i\|^2}$. In this case, $D_\psi(w, w_0)$ is simply $\frac{1}{2}\|w - w_0\|^2$.

Corollary 51. *In particular, for the initialization $w_0 = \arg\min_{w \in \mathbb{R}^p} \psi(w)$, under the conditions of Proposition 50, the SMD iterates converge to*

$$w_\infty = \arg\min_{w \in \mathcal{W}} \psi(w). \tag{9.6}$$

This means that running SMD for a linear model with the aforementioned w_0, without any explicit regularization, results in a solution that has the smallest potential $\psi(\cdot)$ among all solutions, i.e., SMD implicitly regularizes the solution with $\psi(\cdot)$. In particular, this means that SGD initialized around zero acts as an ℓ_2-norm regularizer. In this chapter, we show that these results continue to hold for highly overparameterized nonlinear models in an approximate sense.

9.4.2 Main Results

Let us define

$$D_{L_i}(w, w') := L_i(w) - L_i(w') - \nabla L_i(w')^T (w - w'), \tag{9.7}$$

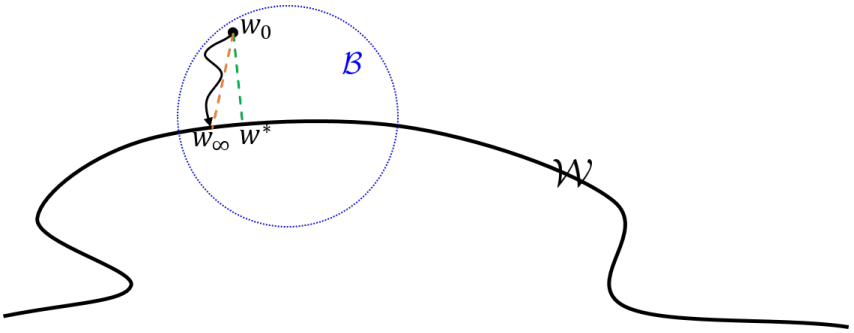


Figure 9.3: An illustration of the parameter space. $\mathcal{W}$ represents the set of global minima, w_0 is the initialization, $\mathcal{B}$ is the local neighborhood, w^* is the closest global minimum to w_0 (in Bregman divergence), and w_∞ is the minimum that SMD converges to.

which is defined in a similar way to a Bregman divergence for the loss function. The difference though is that, unlike the potential function of the Bregman divergence, the loss function $L_i(\cdot) = \ell(y_i - f(x_i, \cdot))$ need not be convex (even when $\ell(\cdot)$ is) due to the nonlinearity of $f(\cdot, \cdot)$.

It has been argued in several recent papers that in highly overparameterized neural networks, because $\mathcal{W}$ is very high-dimensional, any random initialization w_0 is close to it, with high probability [129, 67, 7, 51]. In other words, one can show that, under certain conditions, the distance of a random initialization point w_0 to the manifold scales as

$$D_\psi(\mathcal{W}, w_0)^2 = O(\frac{n}{m}) \tag{9.8}$$

(see the discussion in Section 9.A.4 of the supplementary material). Therefore, in such settings, it is reasonable to make the following assumption about the manifold.

Assumption 1. *Denote the initial point by w_0. There exists $w \in \mathcal{W}$ and a region $\mathcal{B} = \{w' \in \mathbb{R}^p \mid D_\psi(w, w') \leq \epsilon\}$ containing w_0, such that $D_{L_i}(w, w') \geq 0, i = 1, \ldots, n$, for all $w' \in \mathcal{B}$.*

It is important to understand what this assumption means. Since $L_i(\cdot)$ is not necessarily convex, it is certainly not the case that $D_{L_i}(w, w') \geq 0$ for all w'. However, since w is a minimizer of $L_i(\cdot)$, there will be a neighborhood around it such that for all w' in this neighborhood $D_{L_i}(w, w') \geq 0$ (see Fig. 9.4 for an illustration). What we are requiring is that the initialization w_0 be inside the intersection of all such neighborhoods for $i = 1, \ldots, n$. In other words, we require a w_0 close enough to $\mathcal{W}$.

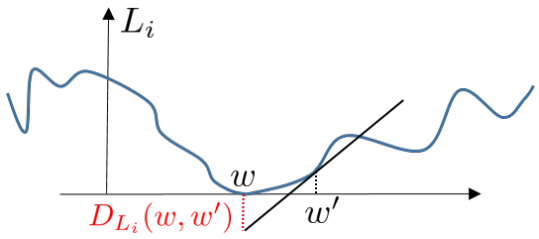


Figure 9.4: An illustration of $D_{L_i}(w, w') \geq 0$ in a local region in Assumption 1.

Our second assumption states that in this local region, the first and second derivatives of the model are bounded.

Assumption 2. *Consider the region $\mathcal{B}$ in Assumption 1. $f_i(\cdot)$ have bounded gradient and Hessian on the convex hull of $\mathcal{B}$, i.e., $\|\nabla f_i(w')\| \leq \gamma$, and $\alpha \leq \lambda_{\min}(H_{f_i}(w')) \leq \lambda_{\max}(H_{f_i}(w')) \leq \beta, i = 1, \ldots, n$, for all $w' \in \operatorname{conv} \mathcal{B}$.*

This is a mild assumption, which is assumed in other related work such as [163] as well. Note that we do *not* require α to be positive (just its boundedness). The following theorem states that under Assumption 1, SMD converges to a global minimum.

Theorem 52. *Consider the set of interpolating parameters $\mathcal{W} = \{w \in \mathbb{R}^m \mid f(x_i, w) = y_i, i = 1, \ldots, n\}$, and the SMD iterates given in (9.3), where every data point is revisited after some steps. Under Assumption 1, for sufficiently small step size, i.e., for any $\eta > 0$ for which $\psi(\cdot) - \eta L_i(\cdot)$ is strictly convex on $\mathcal{B}$ for all i, the following holds.*

1. *All the iterates $\{w_i\}$ remain in $\mathcal{B}$.*
2. *The iterates converge (to w_∞).*
3. *$w_\infty \in \mathcal{W}$.*

Note that, while convergence (to some point) with decaying step size is almost trivial, this result establishes convergence to the solution set with a *fixed* step size. Furthermore, the convergence is *deterministic*, and is not in expectation or with high probability. For example, this result also applies to the case where we cycle through the data deterministically.

We should also remark that the choice of distance in the definition of the "ball" $\mathcal{B}$ was important to be the Bregman divergence with respect to $\psi(\cdot)$ and in that particular order. In fact, one cannot guarantee that the SMD iterates get closer to an interpolating w at every step in the usual Euclidean sense. However, one can establish that it gets closer in $D_\psi(w, \cdot)$. Finally, it is important to note that we need the step size to be small enough to guarantee the strict convexity of $\psi(\cdot) - \eta L_i(\cdot)$ in $\mathcal{B}$, not globally.

Denote the global minimum that is closest to the initialization in Bregman divergence by w^*, i.e.,

$$w^* = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0). \tag{9.9}$$

Recall that in the linear case, this was what SMD converges to. We show that in the nonlinear case, under Assumptions 1 and 2, SMD converges to a point w_∞ which is "very close" to w^*.

Theorem 53. *Define $w^* = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0)$. Under the conditions of Theorem 52, and Assumption 2, the following holds:*

1. $D_\psi(w_\infty, w_0) = D_\psi(w^*, w_0) + o(\epsilon)$,
2. $D_\psi(w^*, w_\infty) = o(\epsilon)$.

In other words, if we start with an initialization that is $O(\epsilon)$ away from $\mathcal{W}$ (in Bregman divergence), we converge to a point $w_\infty \in \mathcal{W}$ that is $o(\epsilon)$ away from w^*. The Bregman divergence of this point is $o(\epsilon)$ from the minimum value it can take.

Corollary 54. *For the initialization $w_0 = \arg\min_{w \in \mathbb{R}^p} \psi(w)$, under the conditions of Theorem 53, $w^* = \arg\min_{w \in \mathcal{W}} \psi(w)$ and the following holds:*

1. $\psi(w_\infty) = \psi(w^*) + o(\epsilon)$,
2. $D_\psi(w^*, w_\infty) = o(\epsilon)$.

9.4.3 Fundamental Identity of SMD

An important tool used in our proofs is a "fundamental identity" that governs the behavior of the iterates of SMD, which holds under very general conditions.

Lemma 55. *For any model $f(\cdot, \cdot)$, any differentiable loss $\ell(\cdot)$, any parameter $w \in \mathcal{W}$, and any step size $\eta > 0$, the following relation holds for the SMD iterates*

$\{w_i\}$:

$$D_\psi(w, w_{i-1}) = D_\psi(w, w_i) + D_{\psi-\eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}), \quad (9.10)$$

for all $i \geq 1$.

This identity allows one to prove the results in a remarkably simple and direct way. The proofs are relegated to the appendix.

The ideas behind this identity are related to H_∞ estimation theory [98, 190], which was originally developed in the 1990s in the context of robust control theory. In fact, it has connections to the minimax optimality of SGD, which was shown by [96] for linear models, and recently extended to nonlinear models and general mirrors by [18].

9.5 Related Work

There have been many efforts in the past few years to study deep learning from an optimization perspective, e.g., [3, 58, 188, 7, 163, 138, 67, 129, 51]. While it is not possible to review all the contributions here, we comment on the ones that are most closely related to ours and highlight the distinctions between our results and those.

Many recent papers have studied the convergence of the (S)GD algorithm in the so-called "overparameterized" setting (or "interpolating" regime), which is common in deep learning [163, 7, 193, 138]. All these works, similar to ours, assume that the initialization is close to the solution space (of global minima), which is a reasonable assumption in highly overparameterized models. However, our results are more general because they extend to SMD.

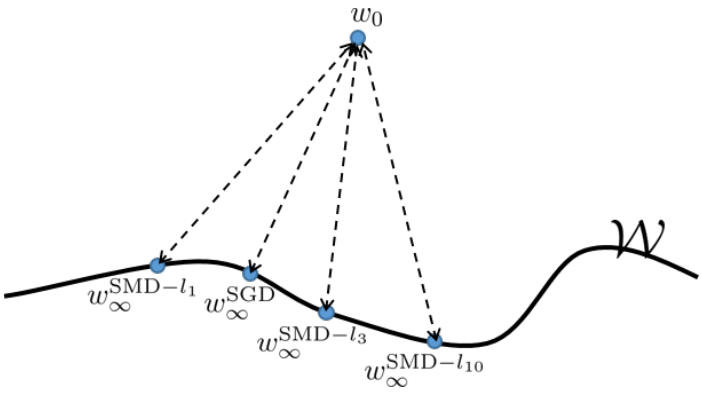


Figure 9.5: An illustration of the experiments in Table 9.1.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	141	9.19×10^3	4.1×10^4	2.34×10^5
2-norm BD	3.15×10^3	562	1.24×10^3	6.89×10^3
3-norm BD	4.31×10^4	107	53.5	1.85×10^2
10-norm BD	6.83×10^{13}	972	7.91×10^{-5}	2.72×10^{-8}

Table 9.1: Fixed Initialization. Distances from final points (global minima) obtained by different algorithms (columns) from the same initialization (Fig. 9.5), measured in different Bregman divergences (rows). First Row: The closest point to w_0 in ℓ_1 Bregman divergence, among the four final points, is exactly the one obtained by SMD with 1-norm potential. Second Row: The closest point to w_0 in ℓ_2 Bregman divergence (Euclidean distance), among the four final points, is exactly the one obtained by SGD. Third Row: The closest point to w_0 in ℓ_3 Bregman divergence, among the four final points, is exactly the one obtained by SMD with 3-norm potential. Fourth Row: The closest point to w_0 in ℓ_{10} Bregman divergence, among the four final points, is exactly the one obtained by SMD with 10-norm potential.

Furthermore, even for the case of SGD, our results are stronger than those in this literature, in the sense that not only do we show convergence to a global minimum, but we also show that the weight vector we converge to, w_∞, say, is close to the interpolating weight vector closest to the initialization, w^*, say. Denoting the initialization by w_0, Oymak and Soltanolkotabi [163] showed that for SGD, $\|w_\infty - w_0\|$ is bounded by a constant factor of $\|w^* - w_0\|$. Our Theorem 53 shows the much stronger statement that $\|w_\infty - w_0\| = \|w^* - w_0\| + o(\|w^* - w_0\|)$. We further show that w_∞ and w^* are very close to one another, viz. $\|w_\infty - w^*\|^2 = o(\|w^* - w_0\|)$, something that could not be inferred from the previous work.

There exist a number of results that characterize the implicit regularization properties of different algorithms in different contexts [154, 137, 87, 85, 194, 86, 18, 142]. The closest ones to our results, since they concern mirror descent, are the works of [85, 18]. The authors in [85] consider *linear* overparameterized models, and show that *if* SMD happens to converge to a global minimum, then that global minimum will be the one that is closest in Bregman divergence to the initialization, a result they obtain by examining the KKT conditions. However, they do not provide any conditions for convergence and whether SMD converges with a fixed step size or not. [18] also study linear models, but derive conditions on the step size for which SMD converges to the aforementioned global minimum. Our results extend the aforementioned to *nonlinear* overparametrized models, and show that, for small enough *fixed* step size, and for initializations close enough to the space of interpolating solutions, SMD

converges to a global minimum, something which had not been shown in any of the previous work. Assuming every data point is revisited often enough, the convergence we establish is *deterministic*. Finally, we show that the solution we converge to exhibits approximate implicit regularization, something that was not known for nonlinear models.

9.6 Experimental Results

In this section, we evaluate the theoretical claims by running systematic experiments for different initializations and different mirrors and computing the distances between the global minima achieved and the initializations, in different Bregman divergences.

While accessing all the points on $\mathcal{W}$ and finding the closest one is impossible, we design systematic experiments to test this claim. We run experiments on some standard deep learning problems, namely, a standard CNN on MNIST [120] and the ResNet-18 [101] on CIFAR-10 [117]. We train the models from different initializations, and with different mirror descents from each particular initialization, until we reach 100% training accuracy, i.e., a point on $\mathcal{W}$. We randomly initialize the parameters of the networks around zero. We choose 6 independent initializations for the CNN, and 8 for ResNet-18, and for each initialization, we run different SMD algorithms with the following four potential functions: (a) ℓ_1 norm, (b) ℓ_2 norm (which is SGD), (c) ℓ_3 norm, and (d) ℓ_{10} norm (as a surrogate for ℓ_∞). See Appendix 9.B for more details on the experiments.

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6	Final 7	Final 8
Initial 1	6×10^2	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 2	2.8×10^3	6.1×10^2	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 3	2.8×10^3	2.9×10^3	5.6×10^2	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 4	2.8×10^3	2.9×10^3	2.8×10^3	5.9×10^2	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 5	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	5.7×10^2	2.8×10^3	2.8×10^3	2.8×10^3
Initial 6	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	5.6×10^2	2.8×10^3	2.8×10^3
Initial 7	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	6×10^2	2.8×10^3
Initial 8	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	5.8×10^2

Table 9.2: Fixed Mirror: SGD. Pairwise distances between different initial points and the final points obtained from them by SGD (Fig. 9.6). Row i: The closest final point to the initial point i, among all the eight final points, is exactly the one obtained by the algorithm from initialization i.

We measure the distances between the initializations and the global minima obtained from different mirrors and different initializations, in different Bregman divergences. Table 9.1, and Table 9.2 show some examples among different mirrors and different initializations, respectively. Fig. 9.7 shows the distances between a particular initial

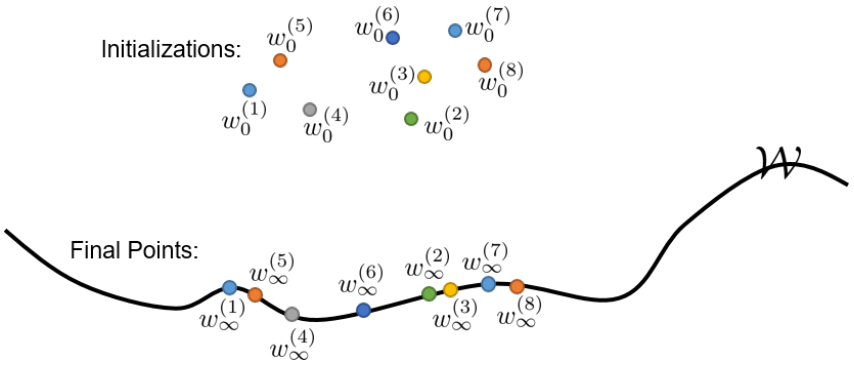
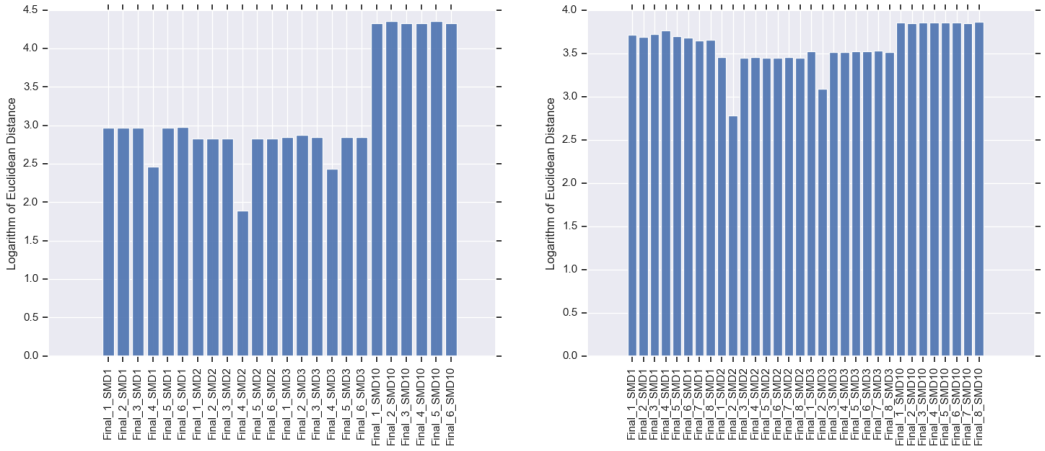


Figure 9.6: An illustration of the experiments in Table 9.2.

point and all the final points obtained from different initializations and different mirrors (the distances are often orders of magnitude different, so we show them in logarithmic scale). The global minimum achieved by any mirror from any initialization is the closest in the correct Bregman divergence, among all mirrors, among all initializations, and among both. This trend is very consistent among all our experiments, which can be found in Appendix 9.B.



MNIST. SGD Starting from Initial 4 CIFAR-10. SGD Starting from Initial 2

Figure 9.7: Distances between a particular initial point and all the final points obtained by both different initializations and different mirrors. The smallest distance, among all initializations and all mirrors, corresponds exactly to the final point obtained from that initial point by SGD. This trend is observed consistently for all other mirror descents and all initializations (see the results in Tables 9.8 and 9.9 in the appendix).

9.7 Conclusion

In this chapter, we studied the convergence and implicit regularization properties of the family of stochastic mirror descent (SMD) for highly overparameterized nonlinear models. From a theoretical perspective, we showed that, under reasonable assumptions, SMD with sufficiently small step size (1) converges to a global minimum and (2) the global minimum converged to is approximately the closest to the initialization in Bregman divergence sense. Furthermore, our extensive experimental results, on various initializations, various mirror descents, and various Bregman divergences, revealed that this phenomenon indeed happens in deep learning, and the solution SMD converges to is the closest to the initialization in Bregman divergence corresponding to that mirror. This further implies that different mirror descent algorithms act as different regularizers, a property that is referred to as *implicit regularization*. The fact that the ℓ_∞-regularized solution showed a better generalization performance than the other ones, while ℓ_1 was the opposite, suggests the importance of a comprehensive study of the role of regularization, and the choice of the best regularizer, to improve the generalization performance of deep neural networks.

9.A Proofs of the Theoretical Results

In this section, we prove the main theoretical results. The proofs are based on a fundamental identity about the iterates of SMD, which holds for all mirrors and all overparametereized (even nonlinear) models (Lemma 55). We first prove this identity, and then use it to prove the convergence and implicit regularization results.

9.A.1 Fundamental Identity of SMD

Let us prove the fundamental identity.

Lemma 55. *For any model $f(\cdot,\cdot)$, any differentiable loss $\ell(\cdot)$, any parameter $w \in \mathcal{W}$, and any step size $\eta > 0$, the following relation holds for the SMD iterates $\{w_i\}$*

$$D_\psi(w, w_{i-1}) = D_\psi(w, w_i) + D_{\psi-\eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}), \quad (9.10)$$

for all $i \geq 1$.

Proof of Lemma 55. Let us start by expanding the Bregman divergence $D_\psi(w, w_i)$ based on its definition

$$D_\psi(w, w_i) = \psi(w) - \psi(w_i) - \nabla\psi(w_i)^T(w - w_i).$$

By plugging the SMD update rule $\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta\nabla L_i(w_{i-1})$ into this, we can write it as

$$D_\psi(w, w_i) = \psi(w) - \psi(w_i) - \nabla\psi(w_{i-1})^T(w - w_i) + \eta\nabla L_i(w_{i-1})^T(w - w_i). \quad (9.11)$$

Using the definition of Bregman divergence for (w, w_{i-1}) and (w_i, w_{i-1}), i.e., $D_\psi(w, w_{i-1}) = \psi(w) - \psi(w_{i-1}) - \nabla\psi(w_{i-1})^T(w - w_{i-1})$ and $D_\psi(w_i, w_{i-1}) = \psi(w_i) - \psi(w_{i-1}) - \nabla\psi(w_{i-1})^T(w_i - w_{i-1})$, we can express this as

$$\begin{aligned} D_\psi(w, w_i) &= D_\psi(w, w_{i-1}) + \psi(w_{i-1}) + \nabla\psi(w_{i-1})^T(w - w_{i-1}) - \psi(w_i) \\ &\qquad - \nabla\psi(w_{i-1})^T(w - w_i) + \eta\nabla L_i(w_{i-1})^T(w - w_i) \quad (9.12) \\ &= D_\psi(w, w_{i-1}) + \psi(w_{i-1}) - \psi(w_i) + \nabla\psi(w_{i-1})^T(w_i - w_{i-1}) \\ &\qquad + \eta\nabla L_i(w_{i-1})^T(w - w_i) \quad (9.13) \\ &= D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta\nabla L_i(w_{i-1})^T(w - w_i). \quad (9.14) \end{aligned}$$

Expanding the last term using $w - w_i = (w - w_{i-1}) - (w_i - w_{i-1})$, and following the definition of $D_{L_i}(.,.)$ from (9.7) for (w, w_{i-1}) and (w_i, w_{i-1}), we have

$$D_\psi(w, w_i) = D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta \nabla L_i(w_{i-1})^T (w - w_{i-1}) - \eta \nabla L_i(w_{i-1})^T (w_i - w_{i-1}) \tag{9.15}$$

$$= D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta \left(L_i(w) - L_i(w_{i-1}) - D_{L_i}(w, w_{i-1})\right) - \eta \left(L_i(w_i) - L_i(w_{i-1}) - D_{L_i}(w_i, w_{i-1})\right) \tag{9.16}$$

$$= D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) + \eta \left(L_i(w) - D_{L_i}(w, w_{i-1})\right) - \eta \left(L_i(w_i) - D_{L_i}(w_i, w_{i-1})\right) \tag{9.17}$$

Note that for all $w \in \mathcal{W}$, we have $L_i(w) = 0$. Therefore, for all $w \in \mathcal{W}$

$$D_\psi(w, w_i) = D_\psi(w, w_{i-1}) - D_\psi(w_i, w_{i-1}) - \eta D_{L_i}(w, w_{i-1}) - \eta L_i(w_i) + \eta D_{L_i}(w_i, w_{i-1}). \tag{9.18}$$

Combining the second and the last terms in the right-hand side leads to

$$D_\psi(w, w_i) = D_\psi(w, w_{i-1}) - D_{\psi - \eta L_i}(w_i, w_{i-1}) - \eta D_{L_i}(w, w_{i-1}) - \eta L_i(w_i), \tag{9.19}$$

for all $w \in \mathcal{W}$, which concludes the proof. □

9.A.2 Convergence of SMD to the Interpolating Set

Now that we have proved Lemma 55, we can use it to prove our main results, in a remarkably simple fashion. Let us first prove the convergence of SMD to the set of solutions.

Assumption 1. *Denote the initial point by w_0. There exists $w \in \mathcal{W}$ and a region $\mathcal{B} = \{w' \in \mathbb{R}^m \mid D_\psi(w, w') \leq \epsilon\}$ containing w_0, such that $D_{L_i}(w, w') \geq 0, i = 1, \ldots, n$, for all $w' \in \mathcal{B}$.*

Theorem 52. *Consider the set of interpolating parameters $\mathcal{W} = \{w \in \mathbb{R}^m \mid f(x_i, w) = y_i, i = 1, \ldots, n\}$, and the SMD iterates given in* (9.3)*, where every data point is revisited after some steps. Under Assumption 1, for sufficiently small step size, i.e., for any $\eta > 0$ for which $\psi(\cdot) - \eta L_i(\cdot)$ is strictly convex for all i, the following holds:*

1. *All the iterates $\{w_i\}$ remain in $\mathcal{B}$;*

2. *The iterates converge (to w_∞);*

3. $w_\infty \in \mathcal{W}$.

Proof of Theorem 52. First we show that all the iterates wil remain in $\mathcal{B}$. Recall the identity of SMD from Lemma 55:

$$D_\psi(w, w_{i-1}) = D_\psi(w, w_i) + D_{\psi-\eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}) \quad (9.10)$$

which holds for all $w \in \mathcal{W}$. If w_{i-1} is in the region $\mathcal{B}$, we know that the last term $D_{L_i}(w, w_{i-1})$ is non-negative. Furthermore, if the step size is small enough that $\psi(\cdot) - \eta L_i(\cdot)$ is strictly convex, the second term $D_{\psi-\eta L_i}(w_i, w_{i-1})$ is a Bregman divergence and is non-negative. Since the loss is non-negative, $\eta L_i(w_i)$ is always non-negative. As a result, we have

$$D_\psi(w, w_{i-1}) \geq D_\psi(w, w_i), \quad (9.20)$$

This implies that $D_\psi(w, w_i) \leq \epsilon$, which means w_i is in $\mathcal{B}$ too. Since w_0 is in $\mathcal{B}$, w_1 will be in $\mathcal{B}$, and therefore, w_2 will be in $\mathcal{B}$, and similarly all the iterates will remain in $\mathcal{B}$.

Next, we prove that the iterates converge and $w_\infty \in \mathcal{W}$. If we sum up identity (9.10) for all $i = 1, \ldots, T$, the first terms on the right- and left-hand side cancel each other telescopically, and we have

$$D_\psi(w, w_0) = D_\psi(w, w_T) + \sum_{i=1}^{T} \left[D_{\psi-\eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}) \right]. \quad (9.21)$$

Since $D_\psi(w, w_T) \geq 0$, we have $\sum_{i=1}^{T} \left[D_{\psi-\eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}) \right] \leq D_\psi(w, w_0)$. If we take $T \to \infty$, the sum still has to remain bounded, i.e.,

$$\sum_{i=1}^{\infty} \left[D_{\psi-\eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}) \right] \leq D_\psi(w, w_0). \quad (9.22)$$

Since the step size is small enough that $\psi(\cdot) - \eta L_i(\cdot)$ is strictly convex for all i, the first term $D_{\psi-\eta L_i}(w_i, w_{i-1})$ is non-negative. The second term $\eta L_i(w_i)$ is non-negative because of the non-negativity of the loss. Finally, the last term $D_{L_i}(w, w_{i-1})$ is non-negative because $w_{i-1} \in \mathcal{B}$ for all i. Hence, all the three terms in the summand are non-negative, and because the sum is bounded, they should go to zero as $i \to \infty$. In particular,

$$D_{\psi-\eta L_i}(w_i, w_{i-1}) \to 0 \quad (9.23)$$

implies $w_i \to w_{i-1}$, i.e., convergence ($w_i \to w_\infty$) (Note that the functions $\psi - \eta L_i$ do not go to zero, as there is a fixed number, i.e., n, of them). Further,

$$\eta L_i(w_i) \to 0. \tag{9.24}$$

This implies that all the individual losses are going to zero, and since every data point is being revisited after some steps, all the data points are being fit. Therefore, $w_\infty \in \mathcal{W}$. □

9.A.3 Closeness of the Final Point to the Regularized Solution

In this section, we show that with the additional Assumption 2 (which is equivalent to $f_i(\cdot)$ having bounded Hessian in $\mathcal{B}$), not only do the iterates remain in $\mathcal{B}$ and converge to the set $\mathcal{W}$, but also they converge to a point which is very close to w^* (the closest solution to the initial point, in Bregman divergence). The proof is again based on our fundamental identity for SMD.

Assumption 2. *Consider the region $\mathcal{B}$ in Assumption 1. $f_i(\cdot)$ have bounded gradient and Hessian on the convex hull of $\mathcal{B}$, i.e., $\|\nabla f_i(w')\| \le \gamma$, and $\alpha \le \lambda_{\min}(H_{f_i}(w')) \le \lambda_{\max}(H_{f_i}(w')) \le \beta, i = 1, \dots, n$, for all $w' \in \operatorname{conv} \mathcal{B}$.*

Theorem 53. *Define $w^* = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0)$. Under the assumptions of Theorem 52, and Assumption 2, the following holds:*

1. $D_\psi(w_\infty, w_0) = D_\psi(w^*, w_0) + o(\epsilon)$,
2. $D_\psi(w^*, w_\infty) = o(\epsilon)$.

Proof of Theorem 53. Recall the identity of SMD from Lemma 55:

$$D_\psi(w, w_{i-1}) = D_\psi(w, w_i) + D_{\psi - \eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}) \tag{9.10}$$

which holds for all $w \in \mathcal{W}$. Summing the identity for all $i \ge 1$, we have

$$D_\psi(w, w_0) = D_\psi(w, w_\infty) + \sum_{i=1}^{\infty} \left[D_{\psi - \eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) + \eta D_{L_i}(w, w_{i-1}) \right]. \tag{9.25}$$

for all $w \in \mathcal{W}$. Note that the only terms in the right-hand side which depend on w are the first one $D_\psi(w, w_\infty)$ and the last one $\eta \sum_{i=1}^{\infty} D_{L_i}(w, w_{i-1})$. In what follows, We will argue that, within $\mathcal{B}$, the dependence on w in the last term is weak and therefore w_∞ is close to w^*.

To further spell out the dependence on w in the last term, let us expand $D_{L_i}(w, w_{i-1})$

$$D_{L_i}(w, w_{i-1}) = 0 - L_i(w_{i-1}) - \nabla L_i(w_{i-1})^T(w - w_{i-1}) \tag{9.26}$$

$$= -L_i(w_{i-1}) + \ell'(y_i - f_i(w_{i-1}))\nabla f_i(w_{i-1})^T(w - w_{i-1}) \tag{9.27}$$

for all $w \in \mathcal{W}$, where the first equality comes from the definition of $D_{L_i}(\cdot, \cdot)$ and the fact that $L_i(w) = 0$ for $w \in \mathcal{W}$. The second equality is from taking the derivative of $L_i(\cdot) = \ell(y_i - f_i(\cdot))$ and evaluating it at w_{i-1}.

By Taylor expansion of $f_i(w)$ around w_{i-1} and using Taylor's theorem (Lagrange's mean-value form), we have

$$f_i(w) = f_i(w_{i-1}) + \nabla f_i(w_{i-1})^T(w - w_{i-1}) + \frac{1}{2}(w - w_{i-1})^T H_{f_i}(\hat{w}_i)(w - w_{i-1}), \tag{9.28}$$

for some $\hat{w}_i$ in the convex hull of w and w_{i-1}. Since $f_i(w) = y_i$ for all $w \in \mathcal{W}$, it follows that

$$\nabla f_i(w_{i-1})^T(w - w_{i-1}) = y_i - f_i(w_{i-1}) - \frac{1}{2}(w - w_{i-1})^T H_{f_i}(\hat{w}_i)(w - w_{i-1}), \tag{9.29}$$

for all $w \in \mathcal{W}$. Plugging this into (9.27), we have

$$D_{L_i}(w, w_{i-1}) = -L_i(w_{i-1}) + \ell'(y_i - f_i(w_{i-1}))\left(y_i - f_i(w_{i-1}) - \frac{1}{2}(w - w_{i-1})^T H_{f_i}(\hat{w}_i)(w - w_{i-1})\right) \tag{9.30}$$

for all $w \in \mathcal{W}$. Finally, by plugging this back into the identity (9.25), we have

$$D_\psi(w, w_0) = D_\psi(w, w_\infty) + \sum_{i=1}^{\infty} \Big[D_{\psi - \eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) - \eta L_i(w_{i-1})$$

$$+ \eta \ell'(y_i - f_i(w_{i-1}))\big(y_i - f_i(w_{i-1}) - \frac{1}{2}(w - w_{i-1})^T H_{f_i}(\hat{w}_i)(w - w_{i-1})\big)\Big]. \tag{9.31}$$

for all $w \in \mathcal{W}$. Note that this can be expressed as

$$D_\psi(w, w_0) = D_\psi(w, w_\infty) + C - \sum_{i=1}^{\infty} \frac{1}{2}\eta \ell'(y_i - f_i(w_{i-1}))(w - w_{i-1})^T H_{f_i}(\hat{w}_i)(w - w_{i-1}), \tag{9.32}$$

for all $w \in \mathcal{W}$, where C does not depend on w:

$$C = \sum_{i=1}^{\infty} \left[D_{\psi - \eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i) - \eta L_i(w_{i-1}) + \eta \ell'(y_i - f_i(w_{i-1}))(y_i - f_i(w_{i-1})) \right].$$

From Theorem 52, we know that $w_\infty \in \mathcal{W}$. Therefore, by plugging it into equation (9.32), and using the fact that $D_\psi(w_\infty, w_\infty) = 0$, we have

$$D_\psi(w_\infty, w_0) = C - \sum_{i=1}^{\infty} \frac{1}{2}\eta \ell'(y_i - f_i(w_{i-1}))(w_\infty - w_{i-1})^T H_{f_i}(w_i')(w_\infty - w_{i-1}), \tag{9.33}$$

where w_i' is a point in the convex hull of w_∞ and w_{i-1} (and therefore also in conv $\mathcal{B}$), for all i. Similarly, by plugging w^*, which is also in $\mathcal{W}$, into (9.32), we have

$$D_\psi(w^*, w_0) = D_\psi(w^*, w_\infty) + C - \sum_{i=1}^{\infty} \frac{1}{2}\eta\ell'(y_i - f_i(w_{i-1}))(w^* - w_{i-1})^T H_{f_i}(w_i'')(w^* - w_{i-1}), \tag{9.34}$$

where w_i'' is a point in the convex hull of w^* and w_{i-1} (and therefore also in conv $\mathcal{B}$), for all i. Subtracting the last two equations from each other yields

$$\begin{aligned} D_\psi(w_\infty, w_0) - D_\psi(w^*, w_0) = -D_\psi(w^*, w_\infty) + \sum_{i=1}^{\infty} \frac{1}{2}\eta\ell'(y_i - f_i(w_{i-1}))\cdot \\ \left[(w^* - w_{i-1})^T H_{f_i}(w_i'')(w^* - w_{i-1}) - (w_\infty - w_{i-1})^T H_{f_i}(w_i')(w_\infty - w_{i-1})\right]. \end{aligned} \tag{9.35}$$

Note that since all w_i' and w_i'' are in conv $\mathcal{B}$, by Assumption 2, we have

$$\alpha\|w_\infty - w_{i-1}\|^2 \leq (w_\infty - w_{i-1})^T H_{f_i}(w_i')(w_\infty - w_{i-1}) \leq \beta\|w_\infty - w_{i-1}\|^2, \tag{9.36}$$

and

$$\alpha\|w^* - w_{i-1}\|^2 \leq (w^* - w_{i-1})^T H_{f_i}(w_i'')(w^* - w_{i-1}) \leq \beta\|w^* - w_{i-1}\|^2. \tag{9.37}$$

Further, again since all the iterates $\{w_i\}$ are in $\mathcal{B}$, it follows that $\|w_\infty - w_{i-1}\|^2 = O(\epsilon)$ and $\|w^* - w_{i-1}\|^2 = O(\epsilon)$. As a result the difference of the two terms, i.e., $\left[(w^* - w_{i-1})^T H_{f_i}(w_i'')(w^* - w_{i-1}) - (w_\infty - w_{i-1})^T H_{f_i}(w_i')(w_\infty - w_{i-1})\right]$, is also $O(\epsilon)$, and we have

$$D_\psi(w_\infty, w_0) - D_\psi(w^*, w_0) = -D_\psi(w^*, w_\infty) + \sum_{i=1}^{\infty} \eta\ell'(y_i - f_i(w_{i-1}))O(\epsilon). \tag{9.38}$$

Now note that $\ell'(y_i - f_i(w_{i-1})) = \ell'(f_i(w) - f_i(w_{i-1})) = \ell'(\nabla f_i(\tilde{w}_i)^T(w - w_{i-1}))$ for some $\tilde{w}_i \in$ conv $\mathcal{B}$. Since $\|w - w_{i-1}\|^2 = O(\epsilon)$ for all i, and since $\ell(\cdot)$ is differentiable and $f_i(\cdot)$ have bounded derivatives, it follows that $\ell'(y_i - f_i(w_{i-1})) = o(\epsilon)$. Furthermore, the sum is bounded. This implies that $D_\psi(w_\infty, w_0) - D_\psi(w^*, w_0) = -D_\psi(w^*, w_\infty) + o(\epsilon)$, or equivalently

$$\left(D_\psi(w_\infty, w_0) - D_\psi(w^*, w_0)\right) + D_\psi(w^*, w_\infty) = o(\epsilon). \tag{9.39}$$

The term in parentheses $D_\psi(w_\infty, w_0) - D_\psi(w^*, w_0)$ is non-negative by definition of w^*. The second term $D_\psi(w^*, w_\infty)$ is non-negative by convexity of ψ. Since both terms are non-negative and their sum is $o(\epsilon)$, each one of them is at most $o(\epsilon)$, i.e.,

$$\begin{cases} D_\psi(w_\infty, w_0) - D_\psi(w^*, w_0) = o(\epsilon) \\ D_\psi(w^*, w_\infty) = o(\epsilon) \end{cases} \tag{9.40}$$

which concludes the proof. □

Corollary 54. *For the initialization* $w_0 = \arg\min_{w\in\mathbb{R}^p} \psi(w)$*, under the conditions of Theorem 53,* $w^* = \arg\min_{w\in\mathcal{W}} \psi(w)$ *and the following holds.*

1. $\psi(w_\infty) = \psi(w^*) + o(\epsilon)$
2. $D_\psi(w^*, w_\infty) = o(\epsilon)$

Proof of Corollary 54. The proof is a straightforward application of Theorem 53. Note that we have

$$D_\psi(w, w_0) = \psi(w) - \psi(w_0) - \nabla\psi(w_0)^T(w - w_0) \tag{9.41}$$

for all w. When $w_0 = \arg\min_{w\in\mathbb{R}^p} \psi(w)$, it follows that $\nabla\psi(w_0) = 0$, and

$$D_\psi(w, w_0) = \psi(w) - \psi(w_0). \tag{9.42}$$

In particular, by plugging in w_∞ and w^*, we have $D_\psi(w_\infty, w_0) = \psi(w_\infty) - \psi(w_0)$ and $D_\psi(w^*, w_0) = \psi(w^*) - \psi(w_0)$. Subtracting the two equations from each other yields

$$D_\psi(w_\infty, w_0) - D_\psi(w^*, w_0) = \psi(w_\infty) - \psi(w^*), \tag{9.43}$$

which, along with the application of Theorem 53, concludes the proof. □

9.A.4 Closeness to the Interpolating Set in Highly Overparameterized Models

As we mentioned earlier, it has been argued in a number of recent papers that for highly overparameterized models, any random initial point is, w.h.p., close to the solution set $\mathcal{W}$ [18, 129, 67, 7, 51]. In the highly overparameterized regime, $m \gg n$, and so the dimension of the manifold $\mathcal{W}$, which is $m - n$, is very large. For simplicity, we outline an argument for the case of Euclidean distance, bearing in mind that a similar argument can be used for general Bregman divergence. Note that the distance of an arbitrarily chosen w_0 to $\mathcal{W}$ is given by

$$\begin{aligned} \min_{w} \quad & \|w - w_0\|^2 \\ \text{s.t.} \quad & y = f(x, w) \end{aligned}$$

where $y = \text{vec}(y_i, i = 1, \ldots, n)$ and $f(x, w) = \text{vec}(f(x_i, w), i = 1, \ldots, n)$. This can be approximated by

$$\begin{aligned} \min_{w} \quad & \|w - w_0\|^2 \\ \text{s.t.} \quad & y \approx f(x, w_0) + \nabla f(x, w_0)^T(w - w_0) \end{aligned}$$

where $\nabla f(x, w_0)^T = \text{vec}(\nabla f(x_i, w)^T, i = 1, \ldots, n)$ is the $n \times m$ Jacobian matrix. The latter optimization can be solved to yield

$$\|w^* - w_0\|^2 \approx (y - f(x, w_0))^T \left(\nabla f(x, w_0)^T \nabla f(x, w_0)\right)^{-1} (y - f(x, w_0)) \quad (9.44)$$

Note that $\nabla f(x, w_0)^T \nabla f(x, w_0)$ is an $n \times n$ matrix consisting of the sum of m outer products. When the x_i are sufficiently random, and $m \gg n$, it is not unreasonable to assume that w.h.p.

$$\lambda_{\min}\left(\nabla f(x, w_0)^T \nabla f(x, w_0)\right) = \Omega(m),$$

from which we conclude

$$\|w^* - w_0\|^2 \approx \|y - f(x, w_0)\|^2 \cdot O(\frac{1}{m}) = O(\frac{n}{m}), \quad (9.45)$$

since $y - f(x, w_0)$ is n-dimensional. The above implies that w_0 is close to w^* and hence $\mathcal{W}$.

9.B More Details on the Experimental Results

In order to evaluate the claim, we run systematic experiments on some standard deep learning problems.

Datasets. We use the standard MNIST [120] and CIFAR-10 [117] datasets.

Architectures. For MNIST, we use a 4-layer convolutional neural network (CNN) with 2 convolution layers and 2 fully connected layers. The convolutional layers and the fully connected layers are picked wide enough to obtain 2×10^6 trainable parameters. Since MNIST dataset has 60,000 training samples, the number of parameters is significantly larger than the number of training data points, and the problem is highly overparameterized. For the CIFAR-10 dataset, we use the standard ResNet-18 [101] architecture without any modifications. CIFAR-10 has 50,000 training samples and with the total number of 11×10^6 parameters in ResNet-18, the problem is again highly overparameterized.

Loss Function. We use the cross-entropy loss as the loss function in our training. We train the models from different initializations, and with different mirror descents from each particular initialization, until we reach 100% training accuracy, i.e., until we hit $\mathcal{W}$.

Initialization. We randomly initialize the parameters of the networks around zero ($\mathcal{N}(0, 0.0001)$). We choose 6 independent initializations for the CNN, and 8 for ResNet-18, and for each initialization, we run the following 4 different SMD algorithms.

Algorithms. We use the mirror descent algorithms defined by the norm potential $\psi(w) = \frac{1}{q}\|w\|_q^q$ for the following four different norms: (a) ℓ_1 norm, i.e., $q = 1 + \epsilon$, (b) ℓ_2 norm, i.e., $q = 2$ (which is SGD), (c) ℓ_3 norm, i.e., $q = 3$, (d) ℓ_{10} norm, i.e., $q = 10$ (as a surrogate for ℓ_∞ norm). The update rule can be expressed as follows.

$$w_{i,j} = \left||w_{i-1,j}|^{q-1}\,\mathrm{sign}(w_{i-1,j}) - \eta\nabla L_i(w_{i-1})_j\right|^{\frac{1}{q-1}} \cdot \mathrm{sign}\left(|w_{i-1,j}|^{q-1}\,\mathrm{sign}(w_{i-1,j}) - \eta\nabla L_i(w_{i-1})_j\right), \quad (9.46)$$

where $w_{i-1,j}$ denotes the j-th element of the w_{i-1} vector.

We use a fixed step size η. The step size is chosen to obtain convergence to global minima.

9.B.1 MNIST Experiments

9.B.1.1 Closest Minimum for Different Mirror Descents with Fixed Initialization

We provide the distances from final points (global minima) obtained by different algorithms from the same initialization, measured in different Bregman divergences for MNIST classification task using a standard CNN. Note that, in all the tables, the smallest element in each row is on the diagonal, which means the point achieved by each mirror has the smallest Bregman divergence to the initialization corresponding to that mirror, among all mirrors. Tables 9.3, 9.4, 9.5, 9.6, 9.7, and 9.8 depict these results for 6 different initializations. The rows are the distance metrics used as the Bregman Divergences with specified potentials. The columns are the global minima obtained using specified SMD algorithms.

Table 9.3: MNIST Initial Point 1.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	2.767	937.8	1.05×10^4	1.882×10^5
2-norm BD	301.6	58.61	261.3	2.118×10^4
3-norm BD	1720	37.45	7.143	2518
10-norm BD	7.453×10^8	773.4	0.2939	0.003545

Table 9.4: MNIST Initial Point 2.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	2.78	945	1.37×10^4	2.01×10^5
2-norm BD	292	59.3	374	2.29×10^4
3-norm BD	1.51×10^3	38.6	11.6	2.71×10^3
10-norm BD	1.06×10^8	831	0.86	0.00321

Table 9.5: MNIST Initial Point 3.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	3.02	968	1.06×10^4	1.9×10^5
2-norm BD	291	60.9	272	2.12×10^4
3-norm BD	1.49×10^3	39.1	7.82	2.49×10^3
10-norm BD	1.1×10^8	900	0.411	0.00318

Table 9.6: MNIST Initial Point 4.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	2.78	1.21×10^3	1.08×10^4	1.92×10^5
2-norm BD	291	77.3	271	2.15×10^4
3-norm BD	1.48×10^3	49.7	7.56	2.52×10^3
10-norm BD	9.9×10^7	1.72×10^3	0.352	0.00296

Table 9.7: MNIST Initial Point 5.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	2.79	958	1.08×10^4	2×10^5
2-norm BD	292	60.4	271	2.28×10^4
3-norm BD	1.49×10^3	39	7.52	2.69×10^3
10-norm BD	9.09×10^7	846	0.342	0.00309

Table 9.8: MNIST Initial Point 6.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	2.96	930	1.08×10^4	1.9×10^5
2-norm BD	308	59	271	2.12×10^4
3-norm BD	1.63×10^3	38.6	7.46	2.47×10^3
10-norm BD	1.65×10^8	864	0.334	0.00295

9.B.1.2 Closest Minimum for Different Initializations with Fixed Mirror

We provide the pairwise distances between different initial points and the final points (global minima) obtained by using fixed SMD algorithms in MNIST dataset using a standard CNN. Note that the smallest element in each row is on the diagonal, which means the closest final point to each initialization, among all the final points, is the one corresponding to that point. Tables 9.9, 9.10, 9.11, and 9.12 depict these results for 4 different SMD algorithms. The rows are the initial points, and the columns are the final points corresponding to each initialization.

9.B.1.3 Closest Minimum for Different Initializations and Different Mirrors

Now we assess the pairwise distances between different initial points and final points (global minima) obtained by all different initializations and all different mirrors (Table 9.8). The smallest element in each row is exactly the final point obtained by that mirror from that initialization, among all the mirrors and all the initial points.

Table 9.9: MNIST 1-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 1-norm.

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6
Initial Point 1	2.7671	20311	20266	20331	20340	20282
Initial Point 2	20332	2.7774	20281	20299	20312	20323
Initial Point 3	20319	20312	3.018	20344	20309	20322
Initial Point 4	20339	20279	20310	2.781	20321	20297
Initial Point 5	20347	20317	20273	20316	2.7902	20311
Initial Point 6	20344	20323	20340	20318	20321	2.964

Table 9.10: MNIST 2-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 2-norm (SGD).

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6
Initial Point 1	58.608	670.75	667.03	684.18	671.36	667.84
Initial Point 2	669.84	59.315	669.16	682.04	669.45	669.98
Initial Point 3	666.35	670.22	60.858	683.44	667.57	669.99
Initial Point 4	669.71	668.86	671.19	77.275	670.33	669.7
Initial Point 5	671.1	669.12	668.45	683.61	60.39	666.04
Initial Point 6	669.46	670.92	671.59	684.32	667.37	59.043

Table 9.11: MNIST 3-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 3-norm.

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6
Initial Point 1	7.143	35.302	32.077	32.659	32.648	32.309
Initial Point 2	32.507	11.578	32.256	32.325	32.225	32.46
Initial Point 3	31.594	34.643	7.8239	32.521	31.58	32.519
Initial Point 4	32.303	34.811	32.937	7.5589	32.617	32.284
Initial Point 5	32.673	34.678	32.071	32.738	7.5188	31.558
Initial Point 6	32.116	34.731	32.376	32.431	31.699	7.4593

Table 9.12: MNIST 10-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 10-norm.

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6
Initial Point 1	0.00354	0.37	0.403	0.286	0.421	0.408
Initial Point 2	0.33	0.00321	0.369	0.383	0.415	0.422
Initial Point 3	0.347	0.318	0.00318	0.401	0.312	0.406
Initial Point 4	0.282	0.38	0.458	0.00296	0.491	0.376
Initial Point 5	0.405	0.418	0.354	0.484	0.00309	0.48
Initial Point 6	0.403	0.353	0.422	0.331	0.503	0.00295

	F1 SMD 1	F2 SMD 1	F3 SMD 1	F4 SMD 1	F5 SMD 1	F6 SMD 1	F1 SMD 2	F2 SMD 2	F3 SMD 2	F4 SMD 2	F5 SMD 2	F6 SMD 2	F1 SMD 3	F2 SMD 3	F3 SMD 3	F4 SMD 3	F5 SMD 3	F6 SMD 3	F1 SMD 10	F2 SMD 10	F3 SMD 10	F4 SMD 10	F5 SMD 10	F6 SMD 10
I1 1-norm BD	2.767105	20310.58	20266.27	20330.6	20340.2	20281.51	937.7902	20501.09	20453.6	20615.37	20505.63	20451.42	10500.44	24298.6	22690.41	22883.13	22928.17	22930.01	188233.4	200749.8	189599.3	192017.6	200332.6	189842.1
I2 1-norm BD	20332.47	2.777443	20280.59	20298.8	20312	20322.66	20477.15	944.8926	20467.58	20572.54	20486.79	20481.46	22902.71	13736.89	22683.03	22823.09	22927.75	22951.2	188019	200838.7	189406.7	191694.7	200319.4	189452.9
I3 1-norm BD	20319.38	20312.19	3.018036	20343.8	20308.9	20322.02	20443.74	20487.21	967.6324	20612.98	20486.93	20485.8	22897.06	24300.62	10609.4	22876.31	22901.84	22949.55	187883.2	201071.8	189571	192131.8	199958.1	189571.5
I4 1-norm BD	20338.77	20279.16	20309.78	2.78104	20321.1	20297.36	20476.14	20461.38	20499.16	1214.917	20499.88	20469.45	22910.51	24283.22	22733.45	10756.58	22928.43	22938.72	187740.6	200692.5	189522.4	192082.9	200434.4	189653.4
I5 1-norm BD	20347.03	20317.23	20273.07	20316.4	2.79019	20310.78	20498.73	20496.97	20464.54	20600.07	957.8013	20484.67	22921.69	24335.41	22722.83	22877.07	10812.1	22955.94	188056.5	200743.9	189707.6	192056.4	200478.6	189883
I6 1-norm BD	20343.59	20322.62	20339.82	20318.4	20320.9	2.964027	20493.68	20504.14	20535.06	20590.71	20491.61	930.2714	22926.8	24311.73	22713.74	22837.8	22900.31	10848.27	187959.4	200482.2	189602.3	192052.5	200309.6	189738.7
I1 2-norm BD	301.6218	928.1953	922.246	925.889	929.909	940.8018	58.60796	670.7482	667.0325	684.1751	671.3561	667.8379	261.2823	760.2361	701.1998	706.1766	704.5516	704.0641	21179.18	22902.48	21188.34	21536.64	22803.44	21162.22
I2 2-norm BD	938.5225	291.6223	924.8324	925.561	926.34	944.1569	669.8414	59.31496	669.1617	682.0358	669.4517	669.9774	703.9956	373.9718	702.4789	703.8165	703.9578	705.4268	21164.67	22901.68	21187.33	21523.37	22797.37	21152.21
I3 2-norm BD	936.4615	928.4752	290.902	926.259	924.438	943.7131	666.3494	670.2202	60.85767	683.4393	667.5668	669.9933	700.8777	758.6538	272.0649	705.6848	701.1583	705.155	21164.46	22904.93	21186.56	21536.03	22787.11	21151.72
I4 2-norm BD	938.7566	926.655	926.2601	290.552	928.945	944.0035	669.7086	668.8569	671.186	77.27538	670.3311	669.7023	703.3976	757.9133	704.6345	270.9099	703.842	704.6346	21161.99	22898.71	21186.54	21541.32	22799.92	21152.3
I5 2-norm BD	940.8469	928.4445	923.667	927.336	291.765	939.7045	671.1005	669.1169	668.446	683.6102	60.39	666.0443	705.3977	758.6884	702.3112	705.3715	270.8719	701.3619	21166.8	22898.1	21191.65	21533.87	22805.13	21162.55
I6 2-norm BD	937.93	929.7885	929.404	927.348	925.181	307.5172	669.4556	670.9225	671.5908	684.3248	667.3748	59.04266	702.8038	759.7584	703.6673	705.2996	700.8271	271.1133	21166.69	22894.56	21188.54	21530.77	22796.98	21153.93
I1 3-norm BD	1719.866	1543.515	1516.246	1512.4	1521.08	1656.464	37.45108	67.57934	66.73737	78.02365	67.95686	66.51245	7.14298	35.30229	32.07697	32.65884	32.64842	32.30852	2517.617	2706.617	2491.476	2519.086	2688.245	2470.969
I2 3-norm BD	1751.333	1510.961	1516.163	1514.81	1518.79	1658.074	66.28766	38.64332	66.75334	78.01804	66.94847	67.09068	32.50659	11.57823	32.25632	32.32539	32.2253	32.45956	2516.606	2705.533	2491.199	2518.034	2687.31	2470.926
I3 3-norm BD	1751.98	1544.446	1486.664	1513.27	1517.48	1658.303	65.47958	67.39749	39.096	78.03239	66.49712	67.24052	31.59447	34.64265	7.823877	32.52136	31.58038	32.51863	2517.107	2706.491	2489.415	2519.598	2687.107	2470.339
I4 3-norm BD	1751.523	1543.899	1517.328	1483.49	1522.07	1659.334	66.36948	67.31509	67.59354	49.69977	67.96119	67.20248	32.30269	34.81075	32.93691	7.558935	32.61658	32.28448	2517.248	2706.852	2491.392	2518.947	2687.751	2470.657
I5 3-norm BD	1753.311	1545.901	1516.143	1515.92	1488.06	1657.359	66.56918	67.42434	67.07494	78.55313	39.04714	66.25287	32.67308	34.67835	32.07084	32.73818	7.518829	31.55844	2517.357	2706.916	2491.048	2519.073	2687.064	2471.216
I6 3-norm BD	1751.224	1544.936	1520.698	1514.66	1519.78	1626.957	66.33501	67.47943	67.81073	78.43179	67.07613	38.58941	32.11641	34.73071	32.37629	32.43067	31.69857	7.459286	2517.511	2706.82	2490.098	2518.297	2687.431	2469.509
I1 10-norm BD	7.45E+08	1.06E+08	1.1E+08	9.9E+07	9.1E+07	1.65E+08	773.3514	831.1445	900.464	1718.299	846.4625	864.5718	0.293932	1.233024	0.782131	0.615488	0.748684	0.707943	0.003545	0.370181	0.403135	0.28582	0.421482	0.408148
I2 10-norm BD	7.45E+08	1.06E+08	1.1E+08	9.9E+07	9.1E+07	1.65E+08	773.7523	830.5577	900.2781	1718.625	846.2303	864.6849	0.61333	0.860265	0.732687	0.725046	0.727329	0.727967	0.330493	0.003207	0.368537	0.382603	0.415105	0.422372
I3 10-norm BD	7.45E+08	1.06E+08	1.1E+08	9.9E+07	9.1E+07	1.65E+08	773.8534	831.2133	900.141	1718.575	846.1995	864.7488	0.63865	1.196859	0.410611	0.735479	0.634941	0.718673	0.347069	0.317821	0.00318	0.400619	0.311682	0.405827
I4 10-norm BD	7.45E+08	1.06E+08	1.1E+08	9.9E+07	9.1E+07	1.65E+08	773.8442	831.1647	900.4524	1718.06	846.5443	864.7191	0.585811	1.241436	0.824513	0.351863	0.819103	0.694199	0.281852	0.379772	0.457535	0.002963	0.49113	0.376261
I5 10-norm BD	7.45E+08	1.06E+08	1.1E+08	9.9E+07	9.1E+07	1.65E+08	773.8727	831.1471	900.3779	1718.562	845.8668	864.717	0.691508	1.273044	0.735977	0.814864	0.342117	0.783273	0.40501	0.417533	0.353985	0.483609	0.003094	0.479598
I6 10-norm BD	7.45E+08	1.06E+08	1.1E+08	9.9E+07	9.1E+07	1.65E+08	773.9967	831.1642	900.7065	1718.531	846.6509	864.2966	0.703456	1.22497	0.82352	0.679747	0.840384	0.33448	0.403348	0.352543	0.421798	0.330755	0.503257	0.002948

Figure 9.8: Different Bregman divergences between all the final points and all the initial points for different mirrors in MNIST dataset using a standard CNN. Note that the smallest element in every single row is on the diagonal, which confirms the theoretical results.

9.B.2 CIFAR-10 Experiments

9.B.2.1 Closest Minimum for Different Mirror Descents with Fixed Initialization

We provide the distances from final points (global minima) obtained by different algorithms from the same initialization, measured in different Bregman divergences for CIFAR-10 classification task using ResNet-18. Note that in all the tables, the smallest element in each row is on the diagonal, which means the point achieved by each mirror has the smallest Bregman divergence to the initialization corresponding to that mirror, among all mirrors. Tables 9.13, 9.14, 9.15, 9.16, 9.17, 9.18, 9.19, and 9.20 depict these results for 8 different initializations. The rows are the distance metrics used as the Bregman Divergences with specified potentials. The columns are the global minima obtained using specified SMD algorithms.

Table 9.13: CIFAR-10 Initial Point 1.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	189	9.58×10^3	4.19×10^4	2.34×10^5
2-norm BD	3.12×10^3	597	1.28×10^3	6.92×10^3
3-norm BD	4.31×10^4	119	55.8	1.87×10^2
10-norm BD	1.35×10^{14}	869	6.34×10^{-5}	2.64×10^{-8}

Table 9.14: CIFAR-10 Initial Point 2.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	275	9.86×10^3	4.09×10^4	2.38×10^5
2-norm BD	4.89×10^3	607	1.23×10^3	7.03×10^3
3-norm BD	9.21×10^4	104	53.5	1.88×10^2
10-norm BD	1.17×10^{15}	225	0.000102	2.65×10^{-8}

Table 9.15: CIFAR-10 Initial Point 3.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	141	9.19×10^3	4.1×10^4	2.34×10^5
2-norm BD	3.15×10^3	562	1.24×10^3	6.89×10^3
3-norm BD	4.31×10^4	107	53.5	1.85×10^2
10-norm BD	6.83×10^{13}	972	7.91×10^{-5}	2.72×10^{-8}

Table 9.16: CIFAR-10 Initial Point 4.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	255	9.77×10^3	4.18×10^4	2.36×10^5
2-norm BD	3.64×10^3	594	1.26×10^3	6.96×10^3
3-norm BD	5.5×10^4	116	54	1.87×10^2
10-norm BD	3.74×10^{14}	640	5.33×10^{-5}	2.67×10^{-8}

Table 9.17: CIFAR-10 Initial Point 5.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	113	9.48×10^3	4.15×10^4	2.32×10^5
2-norm BD	2.95×10^3	572	1.27×10^3	6.85×10^3
3-norm BD	3.68×10^4	109	56.2	1.84×10^2
10-norm BD	2.97×10^{13}	151	5.74×10^{-5}	2.61×10^{-8}

Table 9.18: CIFAR-10 Initial Point 6.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	128	9.25×10^3	4.25×10^4	2.34×10^5
2-norm BD	2.71×10^3	558	1.29×10^3	6.89×10^3
3-norm BD	3.34×10^4	104	55.3	1.85×10^2
10-norm BD	2.61×10^{13}	612	4.74×10^{-5}	2.62×10^{-8}

Table 9.19: CIFAR-10 Initial Point 7.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	223	9.76×10^3	4.38×10^4	2.27×10^5
2-norm BD	2.41×10^3	599	1.37×10^3	6.65×10^3
3-norm BD	2.3×10^4	116	61	1.78×10^2
10-norm BD	4.22×10^{12}	679	6.42×10^{-5}	2.55×10^{-8}

Table 9.20: CIFAR-10 Initial Point 8.

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	145	9.37×10^3	4.17×10^4	2.36×10^5
2-norm BD	2.48×10^3	576	1.26×10^3	6.99×10^3
3-norm BD	2.85×10^4	108	54.5	1.89×10^2
10-norm BD	1.81×10^{13}	1.22×10^3	5.2×10^{-5}	2.64×10^{-8}

9.B.2.2 Closest Minimum for Different Initializations with Fixed Mirror

We provide the pairwise distances between different initial points and the final points (global minima) obtained by using fixed SMD algorithms in CIFAR-10 dataset using ResNet-18. Note that the smallest element in each row is on the diagonal, which means the closest final point to each initialization, among all the final points, is the one corresponding to that point. Tables 9.21, 9.22, 9.23, and 9.24 depict these results for 4 different SMD algorithms. The rows are the initial points and the columns are the final points corresponding to each initialization.

Table 9.21: CIFAR-10 1-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 1-norm.

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6	Final 7	Final 8
Initial 1	1.9×10^2	8.1×10^4	8.1×10^4	8.4×10^4	8×10^4	8.2×10^4	7.8×10^4	7.8×10^4
Initial 2	8.1×10^4	2.7×10^2	8.1×10^4	8.3×10^4	8×10^4	8.2×10^4	7.8×10^4	7.9×10^4
Initial 3	8.1×10^4	8.1×10^4	1.4×10^2	8.4×10^4	8×10^4	8.1×10^4	7.8×10^4	7.8×10^4
Initial 4	8.1×10^4	8.1×10^4	8.1×10^4	2.5×10^2	8×10^4	8.2×10^4	7.8×10^4	7.9×10^4
Initial 5	8.1×10^4	8.1×10^4	8.1×10^4	8.3×10^4	1.1×10^2	8.1×10^4	7.8×10^4	7.8×10^4
Initial 6	8.1×10^4	8.1×10^4	8.1×10^4	8.4×10^4	8×10^4	1.3×10^2	7.8×10^4	7.8×10^4
Initial 7	8.1×10^4	8.1×10^4	8.1×10^4	8.4×10^4	8×10^4	8.1×10^4	2.2×10^2	7.8×10^4
Initial 8	8.1×10^4	8.1×10^4	8.1×10^4	8.4×10^4	7.9×10^4	8.1×10^4	7.8×10^4	1.5×10^2

Table 9.22: CIFAR-10 2-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 2-norm (SGD).

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6	Final 7	Final 8
Initial 1	6×10^2	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 2	2.8×10^3	6.1×10^2	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 3	2.8×10^3	2.9×10^3	5.6×10^2	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 4	2.8×10^3	2.9×10^3	2.8×10^3	5.9×10^2	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3
Initial 5	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	5.7×10^2	2.8×10^3	2.8×10^3	2.8×10^3
Initial 6	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	5.6×10^2	2.8×10^3	2.8×10^3
Initial 7	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	6×10^2	2.8×10^3
Initial 8	2.8×10^3	2.9×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	2.8×10^3	5.8×10^2

Table 9.23: CIFAR-10 3-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 3-norm.

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6	Final 7	Final 8
Initial 1	55.844	103.47	103.61	104.05	106.2	105.32	110.88	104.56
Initial 2	105.87	53.455	103.68	104.04	106.31	105.34	110.93	104.58
Initial 3	105.89	103.59	53.527	104.09	106.29	105.35	110.99	104.55
Initial 4	105.83	103.54	103.64	53.978	106.23	105.3	110.87	104.54
Initial 5	105.82	103.55	103.64	104	56.161	105.34	110.88	104.55
Initial 6	105.91	103.6	103.66	104.1	106.28	55.316	110.94	104.55
Initial 7	105.87	103.51	103.67	103.98	106.26	105.25	61.045	104.5
Initial 8	105.77	103.54	103.59	104.04	106.25	105.28	110.88	54.509

Table 9.24: CIFAR-10 10-norm Bregman Divergence Between the Initial Points and the Final Points obtained by SMD 10-norm.

	Final 1	Final 2	Final 3	Final 4	Final 5	Final 6	Final 7	Final 8
Initial 1	2.64×10^{-8}	2.89×10^{-8}	2.99×10^{-8}	2.81×10^{-8}	2.85×10^{-8}	2.82×10^{-8}	2.66×10^{-8}	2.82×10^{-8}
Initial 2	2.79×10^{-8}	2.65×10^{-8}	2.83×10^{-8}	2.83×10^{-8}	2.71×10^{-8}	2.74×10^{-8}	2.69×10^{-8}	2.88×10^{-8}
Initial 3	2.89×10^{-8}	2.87×10^{-8}	2.72×10^{-8}	2.94×10^{-8}	2.84×10^{-8}	2.89×10^{-8}	2.78×10^{-8}	2.94×10^{-8}
Initial 4	2.79×10^{-8}	2.86×10^{-8}	2.92×10^{-8}	2.67×10^{-8}	2.84×10^{-8}	2.81×10^{-8}	2.69×10^{-8}	2.85×10^{-8}
Initial 5	2.76×10^{-8}	2.88×10^{-8}	2.95×10^{-8}	2.93×10^{-8}	2.61×10^{-8}	2.73×10^{-8}	2.66×10^{-8}	2.83×10^{-8}
Initial 6	2.80×10^{-8}	2.76×10^{-8}	2.93×10^{-8}	2.79×10^{-8}	2.76×10^{-8}	2.62×10^{-8}	2.71×10^{-8}	2.85×10^{-8}
Initial 7	2.73×10^{-8}	2.76×10^{-8}	2.82×10^{-8}	2.79×10^{-8}	2.71×10^{-8}	2.77×10^{-8}	2.55×10^{-8}	2.83×10^{-8}
Initial 8	2.73×10^{-8}	2.79×10^{-8}	2.85×10^{-8}	2.78×10^{-8}	2.75×10^{-8}	2.74×10^{-8}	2.73×10^{-8}	2.64×10^{-8}

9.B.2.3 Closest Minimum for Different Initializations and Different Mirrors

Now we assess the pairwise distances between different initial points and final points (global minima) obtained by all different initializations and all different mirrors (Table 9.8). The smallest element in each row is exactly the final point obtained by that mirror from that initialization, among all the mirrors and all the initial points.

	F1 SMD 1	F2 SMD 1	F3 SMD 1	F4 SMD 1	F5 SMD 1	F6 SMD 1	F7 SMD 1	F8 SMD 1	F1 SMD 2	F2 SMD 2	F3 SMD 2	F4 SMD 2	F5 SMD 2	F6 SMD 2	F7 SMD 2	F8 SMD 2	F1 SMD 3	F2 SMD 3	F3 SMD 3	F4 SMD 3	F5 SMD 3	F6 SMD 3	F7 SMD 3	F8 SMD 3	F1 SMD 10	F2 SMD 10	F3 SMD 10	F4 SMD 10	F5 SMD 10	F6 SMD 10	F7 SMD 10	F8 SMD 10
I1 1-norm BD	189.328	81283.3	81471.4	83604.7	79579.4	81574.5	78218.4	78389.4	9580.35	96492.8	95885.6	96458.8	96180.1	95912	96403.9	95970	41925.9	114691	114844	115624	115215	116010	116934	115410	233735	243939	240590	241635	240066	240783	236041	242224.4
I2 1-norm BD	80718.8	274.637	81361.3	83423	79566.4	81521.8	78162.2	78526.3	96199.3	9858.87	95881.2	96432.7	96177.9	96011	96360.3	96073	115489	40871	114857	115444	115270	116042	117056	115479	240760	238156	240430	241386	239851	240885	236146	242360.5
I3 1-norm BD	80753.1	81232	140.651	83587.2	79590.2	81486.7	78193	78409	96271.6	96499.5	9189.09	96431.2	96189.8	95954.8	96437.4	96029.7	115549	114774	40954.3	115560	115315	116047	117093	115429	240778	243890	233638	241525	239917	240999	236223	242303.6
I4 1-norm BD	80851.2	81366.7	81446.9	254.781	79561.5	81544.5	78242.4	78550.7	96292.8	96533.1	95833.1	9770.32	96166.7	95905	96399.4	96071	115538	114746	114886	41757.1	115308	116051	117009	115446	240852	243762	240360	235800	239705	240813	236021	242218.3
I5 1-norm BD	80735.9	81095.3	81265.8	83396	112.929	81449.6	78178.8	78416.2	96233.6	96518.6	95873.1	96389.1	9476.29	95930.1	96421.5	96077.4	115579	114753	114937	115436	41482.2	116134	116895	115460	240672	243753	240605	241462	232340	240884	236114	242169.7
I6 1-norm BD	80610.2	81289.8	81424	83502.5	79528.5	127.523	78185.1	78419	96169.5	96635.1	95860.7	96465.6	96195.5	9252.54	96447	95938.9	115554	114848	114825	115542	115272	42498.8	116984	115293	240769	243866	240463	241597	239922	233795	236173	242261.8
I7 1-norm BD	80712.9	81231.3	81427.1	83570.8	79616.1	81490.4	223.16	78465.8	96193.4	96429.7	95832.9	96458.2	96115.9	95851.2	9758.27	95989.1	115562	114705	114924	115462	115271	115967	43834.6	115374	240605	243785	240599	241628	239886	240793	227206	242312.7
I8 1-norm BD	80733	81342.6	81374.9	83662	79457.7	81452.1	78241.5	145.142	96221.4	96491.7	95916.8	96440.2	96122.5	95852.2	96417.2	9365.13	115506	114698	114829	115547	115293	115973	116956	41691.3	240719	243866	240445	241632	239894	240812	236172	236014.5
I1 2-norm BD	3120.14	6974.61	5249.23	5782.27	5001.56	4812.6	4416.03	4494.71	597.342	2854.45	2810.65	2842.48	2820.28	2807.59	2847.02	2822.15	1280.42	3267.8	3274.47	3295.87	3304.95	3316.2	3388.3	3296.81	6916.86	7306.33	7200.89	7231.33	7181.46	7201.92	7032.73	7276.412
I2 2-norm BD	5192.44	4890.69	5245.16	5780.78	4997.38	4808.08	4409.99	4497.59	2845.99	607.493	2811.68	2843.32	2821.78	2809.34	2846	2824.65	3310.89	1234.3	3276.02	3294.08	3308.61	3317	3390.21	3297.57	7229.1	7029.11	7197.54	7229.07	7178.02	7205.71	7036.86	7281.481
I3 2-norm BD	5188.32	6962.86	3146.41	5779.56	4997.19	4801.63	4411	4490.97	2846.96	2856.09	562.461	2842.86	2822.19	2808.17	2848.66	2824.24	3312.22	3271.6	1238.4	3296.24	3308.08	3317.55	3392.26	3296.88	7229.01	7307.63	6889.44	7230.78	7179.44	7210.22	7037.09	7279.456
I4 2-norm BD	5193.64	6980.54	5251.91	3637.04	5004.97	4815.21	4416.43	4501.13	2846.63	2855.25	2810.84	594.372	2820.12	2806.85	2846.75	2824.84	3311.1	3270.4	3276.34	1260.59	3306.82	3316.69	3388.77	3297.26	7227.94	7304.05	7196.27	6961.64	7174.85	7204.47	7030.67	7276.045
I5 2-norm BD	5184.99	6957.65	5235.89	5770.93	2948.17	4797.47	4405.55	4490.22	2844.86	2856.32	2811.19	2840.89	571.718	2807.17	2846.79	2824.71	3311.24	3271.1	3276.49	3293.23	1271.14	3318.64	3388	3297.79	7226.45	7304.07	7201.92	7230.03	6845.38	7205.5	7034.49	7277.01
I6 2-norm BD	5190.72	6974.64	5251.31	5782.08	4998.22	2710.92	4413.17	4496.44	2845.9	2857.39	2811.76	2843.42	2821.57	557.986	2847.93	2822.11	3312.84	3272.1	3275.54	3296.07	3307.67	1286.14	3389.88	3295.92	7228.18	7307.39	7199.14	7230.83	7177.58	6888.1	7035.99	7278.548
I7 2-norm BD	5191.64	6968.14	5244.83	5780.7	5002.59	4807.16	2407.1	4496.42	2844.65	2853.81	2811.15	2841.67	2819.3	2805.37	599.485	2823.2	3312.49	3269.3	3277.22	3293.05	3307.42	3314.74	1366.29	3295.35	7221.98	7303.57	7200.85	7232.19	7176.33	7205.15	6654.25	7281.245
I8 2-norm BD	5191.97	6975.22	5247.43	5785.94	4995.53	4806.71	4415.62	2475.94	2843.69	2855.67	2811.02	2841.96	2820.31	2805.18	2847.1	575.791	3308.79	3270.4	3274.15	3295.42	3307.34	3315.04	3388.73	1263.73	7224.78	7305.98	7197.26	7233.95	7175.87	7204.39	7035.03	6987.731
I1 3-norm BD	43111	92154.2	43157.5	55093.1	36896.9	33445.9	23038.8	28562.7	119.1	157.976	160.31	169.291	163.148	157.375	169.689	162.119	55.8444	103.47	103.613	104.053	106.201	105.324	110.875	104.558	186.84	199.04	196.541	197.293	195.919	196.22	191.327	199.1721
I2 3-norm BD	43160.3	92104.2	43157	55093	36896.5	33445.6	23038.2	28563	172.928	104.241	160.359	169.353	163.209	157.416	169.69	162.184	105.868	53.455	103.676	104.044	106.313	105.337	110.933	104.575	197.924	188.573	196.48	197.283	195.88	196.337	191.481	199.3378
I3 3-norm BD	43159.3	92152.5	43106.2	55092.4	36896.2	33444.7	23038	28562.2	172.919	158.038	106.548	169.325	163.224	157.389	169.749	162.168	105.891	103.59	53.5269	104.086	106.287	105.354	110.985	104.549	197.914	199.105	185.391	197.332	195.919	196.475	191.44	199.2385
I4 3-norm BD	43159.8	92154.2	43157.3	55041.3	36897.1	33446.1	23038.5	28563	172.89	157.991	160.313	115.535	163.144	157.333	169.674	162.168	105.832	103.54	103.639	53.9784	106.233	105.3	110.87	104.544	197.856	198.994	196.416	187.302	195.801	196.303	191.255	199.1668
I5 3-norm BD	43159	92151.9	43155.9	55091.7	36847	33444.2	23037.4	28562	172.832	158.005	160.305	169.254	109.372	157.31	169.661	162.143	105.818	103.55	103.638	104.002	56.1612	105.343	110.875	104.548	197.836	199.009	196.562	197.275	184.152	196.306	191.368	199.1883
I6 3-norm BD	43160.1	92154.1	43157.6	55092.9	36896.3	33395.6	23038.3	28562.9	172.923	158.052	160.364	169.35	163.205	103.541	169.742	162.125	105.909	103.6	103.664	104.096	106.281	55.3157	110.939	104.554	197.901	199.117	196.51	197.293	195.832	184.944	191.437	199.2369
I7 3-norm BD	43159.8	92153.1	43156.5	55092.5	36896.7	33445.3	22990.5	28562.7	172.853	157.961	160.332	169.264	163.126	157.3	115.924	162.126	105.875	103.51	103.667	103.98	106.255	105.25	61.0451	104.499	197.719	198.984	196.51	197.306	195.81	196.319	178.337	199.3073
I8 3-norm BD	43160	92154	43157.3	55093.1	36896.3	33445.5	23038.5	28514.4	172.807	158.005	160.31	169.275	163.147	157.297	169.677	108.372	105.771	103.54	103.594	104.043	106.255	105.275	110.882	54.5092	197.798	199.066	196.446	197.388	195.808	196.308	191.362	188.6573
I1 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.64E-08	2.89E-08	2.99E-08	2.81E-08	2.85E-08	2.82E-08	2.66E-08	2.82E-08
I2 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.79E-08	2.65E-08	2.83E-08	2.83E-08	2.71E-08	2.74E-08	2.69E-08	2.88E-08
I3 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.89E-08	2.87E-08	2.72E-08	2.94E-08	2.84E-08	2.89E-08	2.78E-08	2.94E-08
I4 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.79E-08	2.86E-08	2.92E-08	2.67E-08	2.84E-08	2.81E-08	2.69E-08	2.85E-08
I5 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.76E-08	2.88E-08	2.95E-08	2.93E-08	2.61E-08	2.73E-08	2.66E-08	2.83E-08
I6 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.80E-08	2.76E-08	2.93E-08	2.79E-08	2.76E-08	2.62E-08	2.71E-08	2.85E-08
I7 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.73E-08	2.76E-08	2.82E-08	2.79E-08	2.71E-08	2.77E-08	2.55E-08	2.83E-08
I8 10-norm BD	1.4E+14	1.2E+15	6.8E+13	3.7E+14	3E+13	2.6E+13	4.2E+12	1.8E+13	868.965	225.129	972.063	640.205	151.146	612.265	678.628	1219.04	6.3E-05	0.0001	7.9E-05	5.3E-05	5.7E-05	4.7E-05	6.4E-05	5.2E-05	2.73E-08	2.79E-08	2.85E-08	2.78E-08	2.75E-08	2.74E-08	2.73E-08	2.64E-08

Figure 9.9: Different Bregman divergences between all the final points and all the initial points for different mirrors in CIFAR-10 dataset using ResNet-18. Note that the smallest element in every single row is on the diagonal, which confirms the theoretical results.

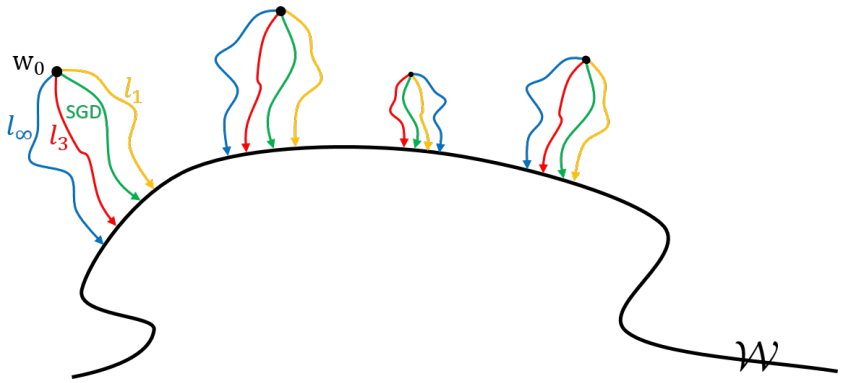


Figure 9.10: An illustration of the experimental results. For each initialization w_0, we ran different SMD algorithms until convergence to a point on the set $\mathcal{W}$ (zero training error). We then measured all the pairwise distances from different w_∞ to different w_0, in different Bregman divergences. The closest point (among all initializations and all mirrors) to any particular initialization w_0 in Bregman divergence with potential $\psi(\cdot) = \|\cdot\|_q^q$ is exactly the point obtained by running SMD with potential $\|\cdot\|_q^q$ from w_0.

9.B.3 Distribution of the Final Weights of the Network

One may be curious to see how the final weights obtained by these different mirrors look, and whether, for example, mirror descent corresponding to the ℓ_1-norm potential induces sparsity. We examine the distribution of the weights in the network for these algorithms starting from the same initialization. Fig. 9.11 shows the histogram of the initial weights, which follows a half-normal distribution. Figs. 9.12 (a), (b), (c), and (d) show the histogram of the weights for ℓ_1-SMD, ℓ_2-SMD (SGD), ℓ_3-SMD, and ℓ_{10}-SMD, respectively. Note that each of the four histograms corresponds to an 11×10^6-dimensional weight vector that perfectly interpolates the data. Even though, perhaps as expected, the weights remain quite small, the histograms are drastically different. The histogram of the ℓ_1-SMD has more weights at and close to zero, which again confirms that it induces sparsity. However, as will be shown in the next section, this is not necessarily good for generalization (in fact, it turns out that ℓ_{10}-SMD has a much better generalization). The histogram of the ℓ_2-SMD (SGD) looks almost identical to the histogram of the initialization, whereas the ℓ_3 and ℓ_{10} histograms are shifted to the right, so much so that almost all weights in the ℓ_{10} solution are non-zero and in the range of 0.005 to 0.04. For comparison, all the distributions are shown together in Fig. 9.12(e).

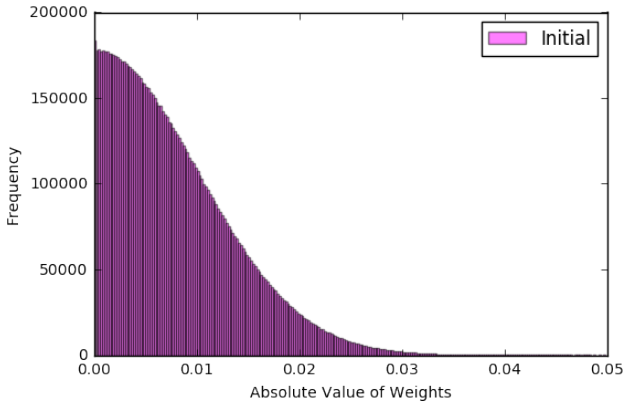


Figure 9.11: Histogram of the absolute value of the initial weights in the network (half-normal distribution).

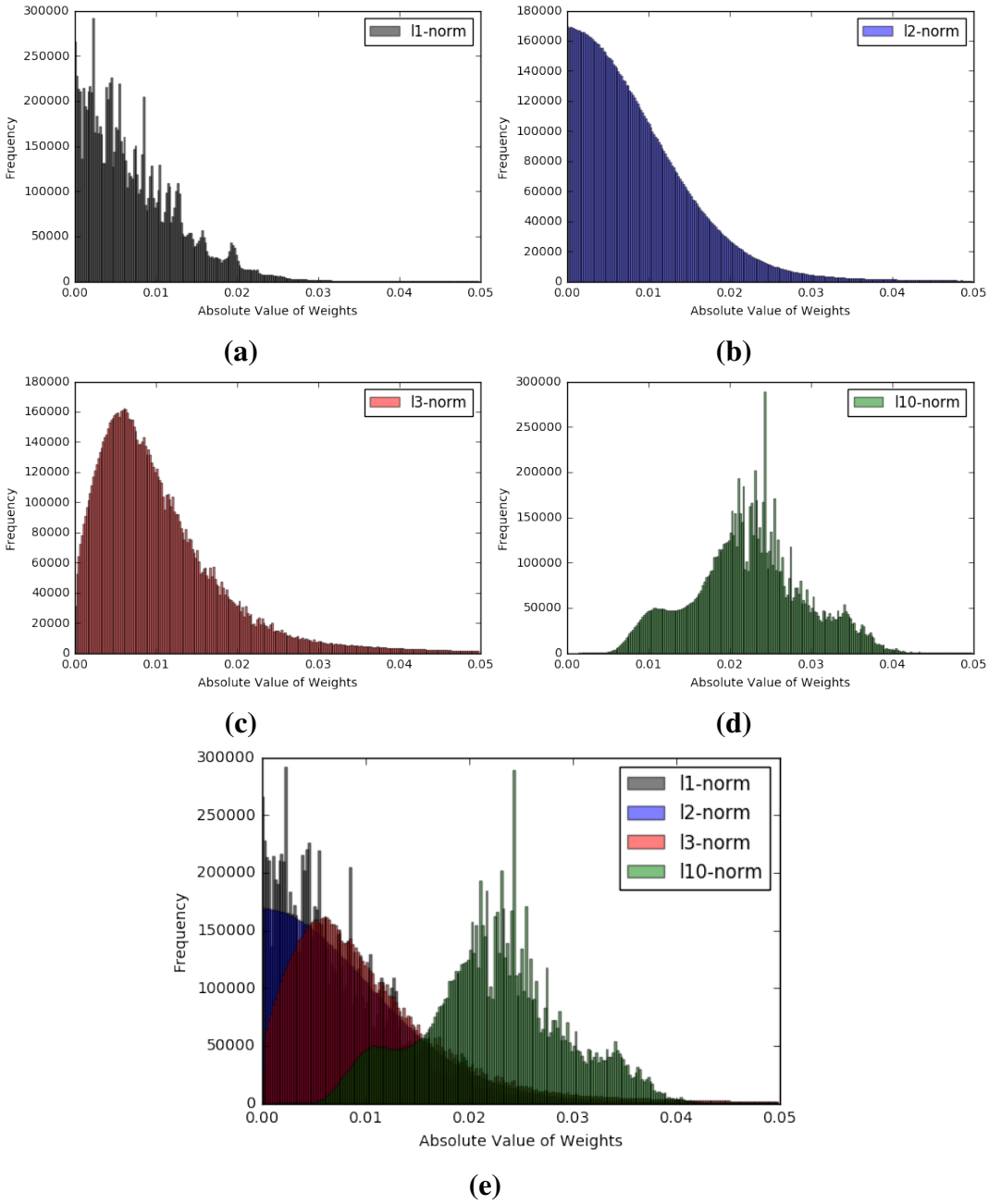


Figure 9.12: Histogram of the absolute value of the final weights in the network for different SMD algorithms: (a) ℓ_1-SMD, (b) ℓ_2-SMD (SGD), (c) ℓ_3-SMD, and (d) ℓ_{10}-SMD. Note that each of the four histograms corresponds to an 11×10^6-dimensional weight vector that perfectly interpolates the data. Even though the weights remain quite small, the histograms are drastically different. ℓ_1-SMD induces sparsity on the weights, as expected. SGD does not seem to change the distribution of the weights significantly. ℓ_3-SMD starts to reduce the sparsity, and ℓ_{10} shifts the distribution of the weights significantly, so much so that almost all the weights are non-zero.

9.B.4 Generalization Errors of Different Mirrors/Regularizers

In this section, we compare the performance of the SMD algorithms discussed before on the test set. This is important for understanding the effect of different regularizers on the generalization of deep networks.

For MNIST, perhaps not surprisingly, all the four SMD algorithms achieve around 99% or higher accuracy. For CIFAR-10, however, there is a significant difference between the test errors of different mirrors/regularizers on the same architecture. Fig. 9.13 shows the test accuracies of different algorithms with eight random initializations around zero, as discussed before. Counter-intuitively, ℓ_{10} performs consistently best, while ℓ_1 performs consistently worse. We should reiterate that the loss function is exactly the same in all these experiments, and all of them have been trained to fit the training set perfectly (zero training error). Therefore, the difference in generalization errors is purely the effect of implicit regularization by different algorithms. This result suggests the importance of a comprehensive study of the role of regularization, and the choice of the best regularizer, to improve the generalization performance of deep neural networks.

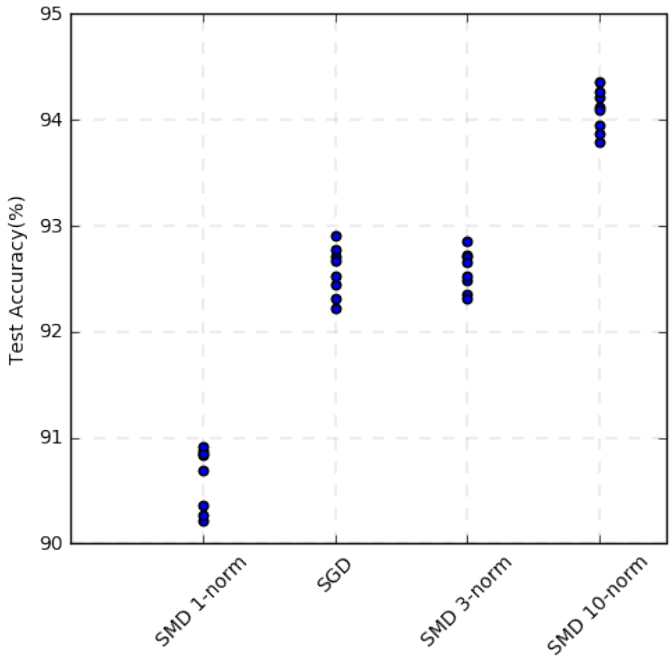


Figure 9.13: Generalization performance of different SMD algorithms on the CIFAR-10 dataset using ResNet-18. ℓ_{10} performs consistently better, while ℓ_1 performs consistently worse.

Chapter 10

STOCHASTIC RESULTS: RISK-SENSITIVE OPTIMALITY AND MEAN-SQUARE CONVERGENCE OF SMD

[1] Navid Azizan and Babak Hassibi. "A Characterization of Stochastic Mirror Descent Algorithms and their Convergence Properties". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 5167–5171. DOI: 10.1109/ICASSP.2019.8682271.

[2] Navid Azizan and Babak Hassibi. "A Stochastic Interpretation of Stochastic Mirror Descent: Risk-Sensitive Optimality". In: *2019 58th IEEE Conference on Decision and Control (CDC)*. 2019, pp. 3960–3965. DOI: 10.1109/CDC40024.2019.9030229.

Stochastic mirror descent (SMD) is a fairly new family of algorithms that has recently found a wide range of applications in optimization, machine learning, and control. It can be considered a generalization of the classical stochastic gradient algorithm (SGD), where instead of updating the weight vector along the negative direction of the stochastic gradient, the update is performed in a "mirror domain" defined by the gradient of a (strictly convex) potential function. This potential function, and the mirror domain it yields, provides considerable flexibility in the algorithm compared to SGD. In this chapter, we exhibit a new interpretation of SMD, namely that it is a risk-sensitive optimal estimator when the unknown weight vector and additive noise are non-Gaussian and belong to the exponential family of distributions. The analysis also suggests a modified version of SMD, which we refer to as symmetric SMD (SSMD). The proofs rely on some simple properties of Bregman divergence, which allow us to extend results from quadratics and Gaussians to certain convex functions and exponential families in a rather seamless way. Furthermore, for vanishing step size, and in the standard stochastic optimization setting, we give a direct and elementary proof for convergence of SMD to the "true" parameter vector which avoids ergodic averaging or appealing to stochastic differential equations.

10.1 Introduction

Stochastic mirror descent (SMD) has become one of the most widely used families of algorithms for optimization, machine learning, and beyond [149, 33, 55, 227, 147, 14, 171], which includes the popular stochastic gradient descent (SGD) as a special

case. The convergence behavior of such algorithms has been extensively studied in the literature [150, 153], under various assumptions. Several other properties and interpretations of SMD have recently been proven in the literature[214, 85]. In earlier work, we have demonstrated a fundamental *conservation law* for SMD and have used it to establish properties such as *minimax optimality*, *deterministic convergence*, and *implicit regularization* [18, 14]. The main contribution of this chapter is to provide a new stochastic interpretation of SMD, i.e., that it is *risk-sensitive optimal*. This generalizes a similar result about SGD in the literature [98, 97]. We also propose a new "more symmetric" version of SMD, called symmetric SMD (SSMD), which is suggested by our analysis.

The chapter is organized as follows. We review the main properties of SMD and the notion of Bregman divergence in Section 10.2. The risk-sensitive optimality result and its proof are provided is Section 10.3. The new SSMD algorithm is presented in Section 10.4. We finally mention another stochastic result about SMD, i.e., its mean-square convergence in the stochastic setting, in Section 10.5, and conclude in Section 10.6.

10.2 Background

Consider a separable loss function of some unknown parameter (or weight) vector $w \in \mathbb{R}^m$:

$$L(w) = \sum_{i=1}^{n} L_i(w),$$

where the $L_i(\cdot)$ are called the instantaneous (or local) loss functions, and where our goal is to minimize $L(\cdot)$ over w. For example, the conventional gradient descent (GD) algorithm can be used as an attempt to perform such minimization. A generalization of GD, called the *mirror descent* (MD) algorithm, was first introduced by Nemirovski and Yudin [149] and can be described as follows. Consider a strictly convex differentiable function $\psi(\cdot)$, called the *potential function*. Then MD is given by the following recursion

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta\nabla L(w_{i-1}), \quad w_0 \tag{10.1}$$

where $\eta > 0$ is known as the step size or learning rate. Note that, due to the strict convexity of $\psi(\cdot)$, the gradient $\nabla\psi(\cdot)$ defines an invertible map so that the recursion in (10.1) yields a unique w_i at each iteration. Compared to classical GD, rather than update the weight vector along the direction of the negative gradient, the update is done in the "mirrored" domain determined by the invertible transformation $\nabla\psi(\cdot)$.

Mirror descent was originally conceived to exploit the geometrical structure of the problem by choosing an appropriate potential. Note that MD reduces to GD when $\psi(w) = \frac{1}{2}\|w\|^2$, since the gradient is simply the identity map. Other examples include the exponentiated gradient descent (also known as the exponential weights) and the p-norms algorithm [82, 78]. As with GD, it is straightforward to show that MD converges to a local minimum of $L(\cdot)$, provided the step size η is small enough.

When n is large, computation of the entire gradient may be cumbersome. Alternatively, in online scenarios, the entire loss function $L(\cdot)$ may not be available and only the local loss functions may be provided at each iteration. In such settings, a stochastic version of MD has been introduced, aptly called *stochastic mirror descent* (SMD), and which can be considered the straightforward generalization of stochastic gradient descent (SGD):

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta\nabla L_i(w_{i-1}), \quad w_0 \tag{10.2}$$

In the offline setting, the various instantaneous loss functions $L_i(\cdot)$ can either be drawn at random, or cycled through periodically. In the online setting, they are provided at each iteration. Unlike MD (and GD), for a fixed step size η, SMD does not generally converge, unless there exists a w that *simultaneously* minimizes every local loss function $L_i(\cdot)$.[1] For this reason, SMD with vanishing learning rate has also been considered

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) - \eta_i\nabla L_i(w_{i-1}), \quad w_0 \tag{10.3}$$

where the learning rate is chosen such that $\eta_i \to 0$. With a vanishing learning rate, it is not surprising that one can attain convergence (since after a while, the algorithm is barely updating the weight vector). What is more interesting is the fact that under suitably decaying rates, one can obtain convergence to a local minimum of $L(\cdot)$ (more on this below).

10.2.1 Bregman Divergence

For any given strictly convex differentiable potential function $\psi(\cdot)$, the *Bregman divergence* is defined as

$$D_\psi(w, w') = \psi(w) - \psi(w') - \nabla\psi(w')^T(w - w'). \tag{10.4}$$

In other words, the Bregman divergence is the difference between the value of the function $\psi(\cdot)$ at a point w and the value of its linear (or first order) approximation

[1] Since if this is not the case, even if the current estimate were at a local minimum of global loss function $L(\cdot)$, w_*, say, any of the local gradients $\nabla L_i(w_*)$ could be nonzero, which would move us away from w_*.

around another point w' (see Fig. 10.1). Since a defining property of a convex function is that its linear approximations always lie below it, we have that $D_\psi(w, w') \geq 0$. Furthermore, since $\psi(\cdot)$ is strictly convex, we have that $D_\psi(w, w') = 0$ iff $w = w'$. Finally, it can be observed that $D_\psi(\cdot, \cdot)$ is convex in its first argument (but not necessarily in the second).

Since the Bregman divergence retains the quadratic (and higher order) terms in the error of the linear approximation of $\psi(w)$ around w', it *inherits many of the properties of quadratics*. For example, the classical "law of cosines,"

$$\|w - w'\|^2 = \|w - w''\|^2 + \|w'' - w'\|^2 - 2(w' - w'')^T(w - w''),$$

generalizes to

$$D_\psi(w, w') = D_\psi(w, w'') + D_\psi(w'', w') - (\nabla\psi(w') - \nabla\psi(w''))^T (w - w''). \quad (10.5)$$

More important for our developments is the following generalization of "completion-of-squares," which we formalize as a lemma.

Lemma 56. *Let $\psi_1(\cdot)$ and $\psi_2(\cdot)$ be strictly convex differentiable functions. Then it holds that*

$$D_{\psi_1}(w, w_1) + D_{\psi_2}(w, w_2) = D_{\psi_1}(w_*, w_1) + D_{\psi_2}(w_*, w_2) + D_{\psi_1+\psi_2}(w, w_*), \quad (10.6)$$

where w_* is the unique solution to the equation

$$\nabla(\psi_1 + \psi_2)(w_*) = \nabla\psi_1(w_1) + \nabla\psi_2(w_2). \quad (10.7)$$

Proof. The identities can be verified by straightforward calculation. The uniqueness of w_* follows from the fact that $\psi_1(\cdot) + \psi_2(\cdot)$ is strictly convex, since it is the sum of two such functions. □

For example, if $\psi(w) = \|w\|^2$ then $D(w, w') = \|w - w'\|^2$, and if $\psi(p) = -H(p)$, where p is a probability vector, then we get that $D_{-H}(p, p') = \sum_i p_i \log \frac{p_i}{p'_i}$ is the KL divergence (or relative entropy).

The last fact about the Bregman divergence that we would like to mention is that a random variable w that has a distribution $w \sim e^{-D_\psi(\cdot, w_0)}$ (i.e., $p(w) = ce^{-D_\psi(w, w_0)}$ for a suitable normalization constant c) is a member of the exponential family of distributions, and satisfies the property

$$\mathbb{E}\nabla\psi(w) = \nabla\psi(w_0). \quad (10.8)$$

In other words, w_0 is the point whose mirror is the mean of the mirror map.

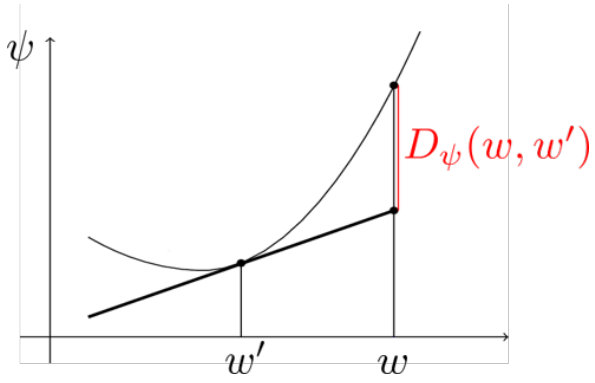


Figure 10.1: Bregman divergence.

10.2.2 Parametric Models

It will now be useful to introduce some parametric models and make our loss functions more explicit. To this end, assume we have a collection of data points

$$\{(x_i, y_i), i = 1, \ldots n\}$$

where $x_i \in \mathbb{R}^d$ is the input and $y_i \in \mathbb{R}$ is the output. We will assume that the pairs (x_i, y_i) are related through some parametric model

$$y_i = f(x_i, w) + v_i, \quad i = 1, \ldots n \tag{10.9}$$

where $f(\cdot, \cdot)$ is a given function and represents the modeling class we are considering, $w \in \mathbb{R}^m$ is the unknown weight vector (or parameter), and v_i represents both measurement noise and modeling errors. In this setting, the global loss function can be written as

$$L(w) = \sum_{i=1}^{n} \underbrace{\ell(y_i, f(x_i, w))}_{L_i(w)}, \tag{10.10}$$

where $\ell(\cdot, \cdot)$ is a (differentiable) local loss function, with the property that $\ell(y_i, f(x_i, w)) = 0$ iff $y_i = f(x_i, w)$. Often $\ell(y_i, f(x_i, w)) = \ell(y_i - f(x_i, w))$, with $\ell(\cdot)$ convex and having a global minimum at zero. In this case,

$$L(w) = \sum_{i=1}^{n} \ell(y_i - f(x_i, w)). \tag{10.11}$$

For example, for quadratic loss we obtain $L(w) = \sum_{i=1}^{n} \frac{1}{2}(y_i - f(x_i, w))^2$. For (10.11), SMD takes the explicit form

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) + \eta \frac{\partial f(x_i, w_{i-1})}{\partial w} \ell'(y_i - f(x_i, w_{i-1})), \quad w_0. \tag{10.12}$$

An important special case is that of linear models

$$y_i = x_i^T w + v_i, \quad i = 1, \ldots, n \tag{10.13}$$

where SMD takes the form

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) + \eta x_i \ell'(y_i - x_i^T w_{i-1}), \quad w_0. \tag{10.14}$$

We will often consider two uncertainties, or error terms, e_i and $e_{p,i}$, defined as follows.

$$e_i := y_i - x_i^T w_{i-1}, \text{ and } e_{p,i} := x_i^T w - x_i^T w_{i-1}.$$

e_i is often referred to as the *innvovations* and is the error in predicting y_i, given the input x_i. $e_{p,i}$ is sometimes called the *prediction error*, since it is the error in predicting the noiseless output $x_i^T w$, i.e., in predicting what the best output of the model is. In the absence of noise, e_i and $e_{p,i}$ coincide.

10.2.3 Local and Global Interpretations of SMD

It is straightforward to show that at each iteration, SMD solves the following optimization problem:

$$w_i = \arg\min_w \; D_\psi(w, w_{i-1}) + \eta w^T \nabla L_i(w_{i-1}), \tag{10.15}$$

which can be verified by setting the gradient of the right hand side of (10.15) to zero. What the above relation shows is that the SMD iterates try to align themselves with the direction of the instantaneous gradient, while also trying to stay close to the previous iterate in Bregman divergence. (The learning rate relatively weights these two objectives.) We refer to (10.15) as the local interpretation of SMD.

We have recently shown that SMD satisfies the following *local conservation law*. [18, 14].

Lemma 57 (Local Conservation Law [18])**.** *Even though the loss function $L_i(w) = \ell(y_i - f(x_i, w))$ may not be convex, define the Bregman divergence $D_{L_i}(w, w')$ in the usual way. Further define the quantity*

$$E_i(w_i, w_{i-1}) := D_{\psi - \eta L_i}(w_i, w_{i-1}) + \eta L_i(w_i). \tag{10.16}$$

Then for each iteration of the SMD updates (10.12), it holds that

$$D_\psi(w, w_{i-1}) + \eta\ell(v_i) = D_\psi(w, w_i) + \eta D_{L_i}(w, w_{i-1}) + E_i(w_i, w_{i-1}). \tag{10.17}$$

Summing the local identities in (10.17) from time 1 to time T leads to the following global conservation law

$$D_\psi(w, w_0) + \eta \sum_{i=1}^{T} \ell(v_i) = D_\psi(w, w_T) + \eta \sum_{i=1}^{T} D_{L_i}(w, w_{i-1}) + \sum_{i=1}^{T} E_i(w_i, w_{i-1}) \tag{10.18}$$

Note that (10.18) holds for *any* horizon T. We refer to it as the global interpretation of SMD. It can be used to show several remarkable *deterministic* properties of the SMD algorithm. We now mention a couple.

Figure 10.2: Local Conservation Law of SMD.

10.2.4 Minimax Optimality of SMD

Using the aforementioned global identity, in [18, 14], the following has been shown.

Theorem 58 (Minimax Optimality [18])**.** *For any T, provided η is small enough so that $\psi(w) - \eta L_i(w)$ is convex for all i, then*

$$\min_{\{w_i\}} \max_{w, \{v_i\}} \frac{D_\psi(w, w_T) + \eta \sum_{i=1}^{T} D_{L_i}(w, w_{i-1})}{D_\psi(w, w_0) + \eta \sum_{i=1}^{T} \ell(v_i)} = 1 \tag{10.19}$$

and SMD with learning rate η is a minimax optimal algorithm achieving the above.

Theorem 58 is a generalization of the H^∞-optimality of the SGD algorithm for linear models and quadratic loss, where it is referred to as LMS [98, 97, 96], to SMD and general models and general losses. When the potential and loss are quadratic, we have $D_\psi(w, w_0) = \|w - w_0\|^2$ and $\ell(v_i) = v_i^2$. The quantity $D_{L_i}(w, w_{i-1}) = (y_i - x_i^T w)^2 - (y_i - x_i^T w_{i-1})^2 + 2x_i^T(w - w_{i-1})(y_i - x_i^T w_{i-1})$, after some simplification, takes on the form

$$D_{L_i}(w, w_{i-1}) = (x_i^T(w - w_{i-1}))^2,$$

which is the square of the so-called *prediction error*. In this case, we recover the H^∞-optimality of LMS, namely that it solves

$$\min_{\{w_i\}} \max_{w, \{v_i\}} \frac{\|w - w_T\|^2 + \eta \sum_{i=1}^{T} (x_i^T(w - w_{i-1}))^2}{\|w - w_0\|^2 + \eta \sum_{i=1}^{T} v_i^2} \tag{10.20}$$

and the optimal value is 1. As mentioned above, Theorem 58 generalizes H^∞-optimality in three ways: it holds for general potential, general loss function, and general nonlinear model.

10.2.5 Convergence and Implicit Regularization

Another interesting property of SMD, which again can be proven using the global conservation law (10.18), is what is referred to as *implicit regularization*. In over-parameterized (underdetermined) models, which are common in compressed sensing and modern deep learning problems, there are (typically a lot) more parameters (unknowns) than data points (measurements). That means there are many parameter vectors (in fact infinitely many) that are consistent with the observations:

$$\mathcal{W} = \left\{ w \in \mathbb{R}^m \mid y_i = x_i^T w,\ i = 1, \dots, n \right\}.$$

The questions of interest in this regime are: (1) does SMD converge to a solution? and (2) if it does so, which solution does it converge to? The following result answers these questions.

Theorem 59 (Convergence to the "Closest" Point [18])**.** *Suppose $l(\cdot)$ is differentiable and convex and has a unique root at 0, $\psi(\cdot)$ is strictly convex, and $\eta > 0$ is such that $\psi - \eta L_i$ is convex for all i. Then for any w_0, the SMD iterates converge to*

$$w_\infty = \arg\min_{w \in \mathcal{W}} D_\psi(w, w_0). \tag{10.21}$$

Corollary 60 (Implicit Regularization [18])**.** *In particular, for the initialization $w_0 = \arg\min_{w \in \mathbb{R}^m} \psi(w)$, under the conditions of Theorem 59, the SMD iterates converge to*

$$w_\infty = \arg\min_{w \in \mathcal{W}} \psi(w). \tag{10.22}$$

This means that running SMD, without any (explicit) regularization, results in a solution that has the smallest potential $\psi(\cdot)$ among all solutions, i.e., SMD implicitly regularizes the solution with $\psi(\cdot)$. In principle, one can choose the potential function for *any* desired convex regularization. For example, we can find the maximum entropy solution by taking the potential to be the negative entropy, or do compressed sensing with $\psi(w) = \|w\|_{1+\epsilon}$ [18, 14].

We should remark that the result extends to quasi-convex losses $\ell(\cdot)$, and it holds locally (in an approximate sense) even for nonlinear models (non-convex cost).

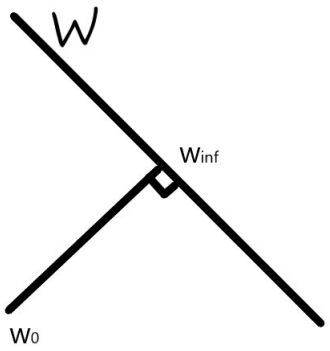


Figure 10.3: w_∞ is the closest solution (among all solutions $\mathcal{W}$) to w_0. Note that this picture is only for the Euclidean distance; in general the "closest" is measured in Bregman divergence.

10.3 Risk-Sensitive Optimality of SMD

The results about SMD discussed in the previous section were deterministic in nature. In this section, we give a stochastic interpretation of SMD, and show that it is risk-sensitive optimal.

Consider a stochastic model $y_i = x_i^T w + v_i, i \geq 1$, where w and $\{v_i\}$ are independent random variables with distributions $w \sim e^{-\frac{1}{\eta} D_\psi(\cdot, w_0)}$ and $v_i \sim e^{-\ell(\cdot)}$, which are members of the exponential family (note that when the potential function $\psi(\cdot)$ and the loss $\ell(\cdot)$ are square, both of these are Gaussian). A conventional quadratic estimator is one that minimizes the expected sum of squared prediction errors, i.e.,

$$\min_{\{z_i\}} \mathbb{E}_{|\{y_i\}} \left[\frac{1}{2} \sum_{i=1}^{T} (x_i^T w - z_i)^2 \right], \tag{10.23}$$

where the expectation is taken over w and $\{v_i\}$ conditioned on the observations, and each z_i in the minimization can only be a function of observations until time $i-1$. For various problems, one may be interested in cost functions more general than quadratic, i.e.,

$$\min_{\{z_i\}} \mathbb{E}_{|\{y_i\}} \left[\sum_{i=1}^{T} D_\ell(y_i - x_i^T w, y_i - z_i) \right]. \tag{10.24}$$

The estimators that solve problems (10.23) and (10.24) are referred to as "risk-neutral" estimators.

An alternative criterion is the "risk-sensitive" (or exponential cost) criterion, which was first introduced in [108] and studied in [196, 210, 195]. In particular, an estimator

that solves the problem

$$\min_{\{z_i\}} \mathbb{E}_{|\{y_i\}} \exp\left(\frac{1}{2}\sum_{i=1}^{T}(x_i^T w - z_i)^2\right), \tag{10.25}$$

is called a "risk-averse" estimator. The reason is that in such a criterion, very large weights are placed on large errors, and hence, the estimator is more concerned about large values of error (their rare occurrence) than the moderate values of error.

Similar as in (10.24), one can consider exponential cost of errors measured with a more general distance than quadratic, i.e.,

$$\min_{\{z_i\}} \mathbb{E}_{|\{y_i\}} \exp\left(\sum_{i=1}^{T} D_\ell(y_i - x_i^T w, y_i - z_i)\right), \tag{10.26}$$

It has been shown in [97, 98] that SGD for square loss (aka LMS) solves the problem (10.25). In other words, LMS is risk-sensitive optimal. Formally, the result is as follows.

Theorem 61 (Hassibi et al. [98])**.** *Consider the model $y_i = x_i^T w + v_i, i \geq 1$, where w and $\{v_i\}$ are independent Gaussian random variables with means w_0 and 0 and variances ηI and I, respectively. Further, suppose that $\{x_i\}$ are persistently exciting and $0 < \eta < \frac{1}{\|x_i\|^2}, \forall i$. Then the solution to the following optimization problem*

$$\min_{\{z_i\}} \mathbb{E}_{|\{y_i\}} \exp\left(\frac{1}{2}\sum_{i=1}^{T}(x_i^T w - z_i)^2\right)$$

where the expectation is taken over w conditioned on the observations, and z_i is only allowed to depend on observations up to time $i-1$, is given by $z_i = x_i^T w_{i-1}$, where $\{w_i\}$ are the SGD iterates.

We should further remark that no larger exponent than $1/2$ is possible (no algorithm can attain a finite cost if the exponent is larger than $1/2$).

The following result generalizes the risk-sensitive optimality of SGD for quadratic errors, to that of SMD for general Bregman-divergence errors.

Theorem 62. *Consider the model $y_i = x_i^T w + v_i, i \geq 1$, where w and $\{v_i\}$ are independent random variables with distributions $w \sim e^{-\frac{1}{\eta}D_\psi(\cdot, w_0)}$ and $v_i \sim e^{-l(\cdot)}$. Further, suppose that $\{x_i\}$ are persistently exciting, and $\psi - \eta L_i$ is strictly convex for all i. Then the solution to the following optimization problem*

$$\min_{\{z_i\}} \mathbb{E}_{|\{y_i\}} \exp\left(\sum_{i=1}^{T} D_\ell(y_i - x_i^T w, y_i - z_i)\right),$$

where the expectation is taken over w conditioned on the observations, and z_i is only allowed to depend on observations up to time $i-1$, is given by $z_i = x_i^T w_{i-1}$, where $\{w_i\}$ are the SMD iterates.

10.3.1 Proof of Theorem 62

The expected exponential cost that needs to be minimized in Theorem 62 is given by

$$C \int \exp\left(-\frac{1}{\eta} D_\psi(w, w_0) - \sum_{i=1}^{T} \ell(y_i - x_i^T w) + \sum_{i=1}^{T} D_\ell(y_i - x_i^T w, y_i - z_i)\right) dw,$$

where C is a normalization constant that guarantees we are integrating the cost against a conditional distribution. The challenge in evaluating the above integral over w is that w appears in all three terms of the exponent. In order to facilitate the computation of this integral, it will be useful to use the completion-of-squares formula of Lemma 56 to gather w into a single term. The following lemma provides precisely what we need.

Lemma 63. *It holds that*

$$\begin{aligned} &-\frac{1}{\eta} D_\psi(w, w_0) - \sum_{i=1}^{T} \ell(y_i - x_i^T w) + \sum_{i=1}^{T} D_\ell(y_i - x_i^T w, y_i - z_i) = \\ &\quad -\frac{1}{\eta} D_\psi(w, w_T) - \sum_{i=1}^{T} \left[\frac{1}{\eta} D_\psi(w_i, w_{i-1}) + \ell(y_i - x_i^T w_i) - D_\ell(y_i - x_i^T w_i, y_i - z_i)\right] \end{aligned}$$

where the w_i, $i = 1, \ldots, T$ are given by the recursion

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) + \eta x_i \ell'(y_i - z_i). \tag{10.27}$$

Proof. The proof is based on telescopically summing the local identity

$$\begin{aligned} &-\frac{1}{\eta} D_\psi(w, w_{i-1}) - \ell(y_i - x_i^T w) + D_\ell(y_i - x_i^T w, y_i - z_i) = \\ &\qquad -\frac{1}{\eta} D_\psi(w, w_i) - \frac{1}{\eta} D_\psi(w_i, w_{i-1}) - \ell(y_i - x_i^T w_i) + D_\ell(y_i - x_i^T w_i, y_i - z_i), \end{aligned}$$

from $i = 1$ to $i = T$, where the w_i are given through the recursion (10.27). This local identity can be either verified directly or obtained through two successive uses of Lemma 56. □

As promised, Lemma 63 gathers w into a single term so that the integral over w can be performed. Once this integral is performed, we are left with the following cost function

$$C' \exp\left(- \sum_{i=1}^{T} \frac{1}{\eta} D_\psi(w_i, w_{i-1}) + \ell(y_i - x_i^T w_i) - D_\ell(y_i - x_i^T w_i, y_i - z_i) \right),$$

where C' is a constant obtained after integrating out w. The above cost function must be recursively minimized over the z_i, which are only allowed to be functions of $\{y_j, j < i\}$, respectively. It is not clear how to do so from the above expression. The next lemma provides an identity that makes this recursive minimization straightforward.

Lemma 64. *It holds that*

$$\ell(y_i - x_i^T w_i) - D_\ell(y_i - x_i^T w_i, y_i - z_i) =$$
$$\ell(y_i - x_i^T w_{i-1}) + \frac{1}{\eta} \left(\nabla\psi(w_i) - \nabla\psi(w_{i-1})\right)^T (w_i - w_{i-1}) - D_\ell(y_i - x_i^T w_{i-1}, y_i - z_i).$$

Proof. This can be verified by perhaps tedious, but straightforward, calculations. □

In view of Lemma 64, the cost function to recursively minimize is

$$C' \exp\Bigg(- \sum_{i=1}^{T} \frac{1}{\eta} D_\psi(w_i, w_{i-1}) + \ell(y_i - x_i^T w_{i-1}) + \frac{1}{\eta} \left(\nabla\psi(w_i) - \nabla\psi(w_{i-1})\right)^T (w_i - w_{i-1}) - D_\ell(y_i - x_i^T w_{i-1}, y_i - z_i) \Bigg).$$

Note that, at any time i, the only term that z_i has control over (in the sense that it is a term that depends only on past y_j) is the term

$$D_\ell(y_i - x_i^T w_{i-1}, y_i - z_i).$$

(The other terms that are influenced by z_i, such as w_i, are influenced also by y_i—see (10.27)—so that z_i cannot knowledgeably minimize them.) The term $D_\ell(y_i - x_i^T w_{i-1}, y_i - z_i)$ can be minimized, and in fact set to zero, by taking

$$z_i = x_i^T w_{i-1}, \tag{10.28}$$

which when plugging into (10.27) yields SMD. *This completes the proof.* (The attentive reader will have noticed that we needed Lemma 64 since it was not clear how to minimize $D_\ell(y_i - x_i^T w_i, y_i - z_i)$ over z_i, because we could not have taken $z_i = x_i^T w_i$ as w_i depends on y_i and z_i is not allowed to.)

10.4 Symmetric SMD (SSMD)

Our proof of the risk-sensitive optimality of SMD has led us to an alternative, and more symmetric version, of the algorithm that we refer to as symmetric SMD (or SSMD) and which may be of independent interest. The SSMD iterations are given by

$$\nabla\psi(w_i) = \nabla\psi(w_{i-1}) + \eta x_i \left(\ell'(y_i) - \ell'(x_i^T w_{i-1})\right), \quad w_0. \tag{10.29}$$

SSMD satisfies the following risk-sensitive optimality.

Theorem 65. *Consider the model $y_i = x_i^T w + v_i, i \geq 1$, where w and $\{v_i\}$ are independent random variables with $w|\{y_i\} \sim e^{-\frac{1}{\eta}D_\psi(\cdot,w_0)-D_\ell(x_i^T\cdot,y_i)}$. Further, suppose that $\{x_i\}$ are persistently exciting, and $\psi - \eta L_i$ is strictly convex for all i. Then the solution to the following optimization problem*

$$\min_{\{z_i\}} \mathbb{E}_{|\{y_i\}} \exp\left(\sum_{i=1}^{T} D_\ell(x_i^T w, z_i)\right),$$

where the expectation is taken over w conditioned on the observations, and z_i is only allowed to depend on observations up to time $i-1$, is given by $z_i = x_i^T w_{i-1}$, where $\{w_i\}$ are the SSMD iterates.

Proof. The proof is similar to that of Theorem 62 and is omitted for brevity. □

We note that the difference between SMD and SSMD is that the noise is now distributed according to $v_i \sim e^{-D_\ell(x_i^T w, y_i)}$, rather than $v_i \sim e^{-\ell(y_i - x_i^T w)}$, and that the exponent of the cost function is $D_\ell(x_i^T w, z_i)$, rather than $D_\ell(y - x_i^T w, y_i - z_i)$. The distributions and costs for SSMD appear to be more natural.

10.5 Mean-Square Convergence of SMD

In the previous sections, we showed several fundamental deterministic and stochastic properties of SMD. One may ask how do these results relate to the conventional mean-square convergence results, such as [150]. It turns out that the fundamental identity (conservation law (10.18)) of SMD allows proving such stochastic convergence results in a direct way (which avoids appealing to stochastic differential equations and ergodic averaging). The time-varying version of the fundamental identity of SMD is as follows.

$$D_\psi(w, w_0) + \sum_{i=1}^{T} \eta_i l(v_i) = D_\psi(w, w_T) + \sum_{i=1}^{T} \left(E_i(w_i, w_{i-1}) + \eta_i D_{L_i}(w, w_{i-1})\right). \tag{10.30}$$

As mentioned before, for vanishing step size, convergence of any algorithm is not surprising, and is in fact trivial (because you are not updating anymore). However, the more interesting question is whether the algorithm converges to anything interesting. It turns out that when the data points are generated according to a stochastic model with white noise, SMD converges to the "true" parameter. More specifically, consider a model $y_i = x_i^T w + v_i, i \geq 1$, where v_i are iid with $\mathbb{E}[v_i] = 0$ and $\mathbb{E}[v_i^2] = \sigma^2$, and the inputs x_i are persistently exciting. Note that this is different from the setting of Theorem 62, in that the noises v_i need not be Gaussian or from the the exponential family (the only assumption is whiteness), and the parameter w is deterministic. One can show that SMD with decaying step size indeed converges to w, under suitable conditions on the step size sequence.

Proposition 66. *Consider $y_i = x_i^T w + v_i, i \geq 1$, where $\mathbb{E}[v_i] = 0$, $\mathbb{E}[v_i v_j] = \sigma^2 \delta_{ij}$, and the x_i are persistently exciting. The stochastic mirror descent iterates for any strongly convex potential $\psi(\cdot)$, for a square loss, converge to w in the mean-square sense, if the step size sequence $\{\eta_i\}$ satisfies $\sum_{i=1}^{\infty} \eta_i = \infty$, $\sum_{i=1}^{\infty} \eta_i^2 < \infty$.*

The step size conditions $\sum_{i=1}^{\infty} \eta_i = \infty$, $\sum_{i=1}^{\infty} \eta_i^2 < \infty$ are known as Robbins–Monro [176] conditions.

Proof. For the square loss and a linear model, after some simple algebra, the identity (10.30) reduces to the following form.

$$D_\psi(w, w_0) = D_\psi(w, w_T) + \sum_{i=1}^{T} \Big(D_\psi(w_i, w_{i-1}) + \eta_i e_{p,i} v_i - \eta_i (e_{p,i} + v_i) x_i^T (w_i - w_{i-1}) + \eta_i e_{p,i}^2 \Big), \quad (10.31)$$

where we have used the fact that $e_i = e_{p,i} + v_i$.

On the other hand, the update rule $\nabla\psi(w_i) = \nabla\psi(w_{i-1}) + \eta_i(e_{p,i} + v_i)x_i$ can be expressed, using a Taylor expansion, as

$$\begin{aligned} w_i &= \nabla\psi^{-1} \left(\nabla\psi(w_{i-1}) + \eta_i (e_{p,i} + v_i) x_i \right) \\ &= w_{i-1} + \eta_i M_i (e_{p,i} + v_i) x_i + O(\eta_i^2), \end{aligned}$$

where $M_i := \nabla^2\psi(w_{i-1})^{-1}$. This implies that $D_\psi(w_i, w_{i-1}) = \frac{1}{2}(w_i - w_{i-1})^T \nabla^2\psi(w_{i-1})(w_i - w_{i-1}) + O(\eta_i^3) = \frac{1}{2}\eta_i^2 (e_{p,i} + v_i)^2 x_i^T M_i x_i + O(\eta_i^3)$. Plugging this into (10.31) leads to

$$D_\psi(w, w_0) = D_\psi(w, w_T) + \sum_{i=1}^{T} \Big(\eta_i e_{p,i} v_i - \frac{1}{2}\eta_i^2 (e_{p,i} + v_i)^2 x_i^T M_i x_i + \eta_i e_{p,i}^2 + O(\eta_i^3) \Big). \quad (10.32)$$

Taking expected values from both sides, noting that $e_{p,i}$ and w_{i-1} are independent of v_i, we get

$$\mathbb{E}[D_\psi(w, w_0)] = \mathbb{E}[D_\psi(w, w_T)] + \sum_{i=1}^{T} \Big(-\frac{\sigma^2}{2}\eta_i^2 \mathbb{E}\left[x_i^T M_i x_i\right] - \frac{1}{2}\eta_i^2 \mathbb{E}\left[e_{p,i}^2 x_i^T M_i x_i\right] + \eta_i \mathbb{E}\left[e_{p,i}^2\right] + O(\eta_i^3)\Big). \quad (10.33)$$

From strong convexity of $\psi(\cdot)$, we have $\nabla^2\psi(w_{i-1}) \geq \alpha I$, and therefore $\mathbb{E}\left[x_i^T M_i x_i\right] \leq \frac{1}{\alpha}\|x_i\|^2$ and $\mathbb{E}\left[e_{p,i}^2 x_i^T M_i x_i\right] \leq \frac{1}{\alpha}\|x_i\|^2 \mathbb{E}\left[e_{p,i}^2\right]$. As a result, we have that $\sum_{i=1}^{T} \eta_i(1 - \frac{\|x_i\|^2}{2\alpha}\eta_i)\mathbb{E}\left[e_{p,i}^2\right] < \infty$ because $\sum_{i=1}^{T} \eta_i^2 < \infty$ and $\sum_{i=1}^{T} O(\eta_i^3) < \infty$, which implies that $\mathbb{E}\left[e_{p,i}^2\right]$ goes to zero. If the inputs are persistently exciting, this implies that $\mathbb{E}\left[\|w - w_{i-1}\|^2\right] \to 0$, which means SMD converges to the true parameter, in mean-square sense. □

10.6 Conclusion

In this chapter, we reviewed several fundamental properties of the stochastic mirror descent (SMD) family of algorithms, and provided a new stochastic interpretation of them, namely, that they are risk-sensitive optimal. We also provided a direct and elementary proof of the stochastic convergence of SMD to the true parameter for vanishing step size in the standard stochastic optimization setting. The risk-sensitive optimality result generalizes a known result in the literature about the special case of SGD (aka LMS). Our analysis inspired a new algorithm, which is a "more symmetric" variant of SMD. Future work may concern studying this new algorithm and its convergence properties in more detail.

BIBLIOGRAPHY

[1] *2015 Top 500 North American Solar Contractors*. http://www.solarpowerworldonline.com.

[2] Daron Acemoglu et al. *Network security and contagion*. Tech. rep. National Bureau of Economic Research, 2013.

[3] Alessandro Achille and Stefano Soatto. "On the emergence of invariance and disentangling in deep representations". In: *arXiv preprint arXiv:1706.01350* (2017).

[4] Hyoung Jun Ahn. "Random propagation in complex systems: nonlinear matrix recursions and epidemic spread". PhD thesis. California Institute of Technology, 2014.

[5] Hyoung Jun Ahn and Babak Hassibi. "Global dynamics of epidemic spread over complex networks". In: *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE. 2013, pp. 4579–4585.

[6] Hyoung Jun Ahn and Babak Hassibi. "On the Mixing Time of the SIS Markov Chain Model for Epidemic Spread". In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE. 2014.

[7] Zeyuan Allen-Zhu et al. "A convergence theory for deep learning via over-parameterization". In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019.

[8] Tansu Alpcan and Tamer Basar. *Network security: A decision and game-theoretic approach*. Cambridge University Press, 2010.

[9] Dario Amodei et al. "Deep speech 2: End-to-end speech recognition in english and mandarin". In: *International Conference on Machine Learning*. 2016, pp. 173–182.

[10] Veronica Araoz and Kurt Jörnsten. "Semi-Lagrangean approach for price discovery in markets with non-convexities". In: *European Journal of Operational Research* 214.2 (2011), pp. 411–417.

[11] Mario Arioli et al. "A block projection method for sparse matrices". In: *SIAM Journal on Scientific and Statistical Computing* 13.1 (1992), pp. 47–70.

[12] D. B. Arnold et al. "Model-Free Optimal Control of VAR Resources in Distribution Systems: An Extremum Seeking Approach". In: *IEEE Transactions on Power Systems* PP.99 (2015), pp. 1–11. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2015.2502554.

[13] Navid Azizan. "Optimization Algorithms for Large-Scale Systems: From Deep Learning to Energy Markets". In: *SIGMETRICS Performance Evaluation Review* 47.3 (2020), pp. 2–5. ISSN: 0163-5999. DOI: 10.1145/3380908.3380910.

[14] Navid Azizan and Babak Hassibi. "A Characterization of Stochastic Mirror Descent Algorithms and their Convergence Properties". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 5167–5171. DOI: 10.1109/ICASSP.2019.8682271.

[15] Navid Azizan and Babak Hassibi. "A Stochastic Interpretation of Stochastic Mirror Descent: Risk-Sensitive Optimality". In: *2019 58th IEEE Conference on Decision and Control (CDC)*. 2019, pp. 3960–3965. DOI: 10.1109/CDC40024.2019.9030229.

[16] Navid Azizan and Babak Hassibi. "SIRS epidemics on complex networks: Concurrence of exact Markov chain and approximated models". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015, pp. 2919–2926. DOI: 10.1109/CDC.2015.7402660.

[17] Navid Azizan and Babak Hassibi. "Stochastic gradient/mirror descent: Minimax optimality and implicit regularization". In: *2018 Neural Information Processing Systems (NeurIPS) Deep Learning Theory Workshop*. 2018.

[18] Navid Azizan and Babak Hassibi. "Stochastic gradient/mirror descent: Minimax optimality and implicit regularization". In: *2019 International Conference on Learning Representations (ICLR)*. 2019.

[19] Navid Azizan et al. "A Study of Generalization of Stochastic Mirror Descent Algorithms on Overparameterized Nonlinear Models". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 3132–3136. DOI: 10.1109/ICASSP40776.2020.9053864.

[20] Navid Azizan et al. "Analysis of exact and approximated epidemic models over complex networks". In: *arXiv preprint arXiv:1609.09565* (2016). URL: http://arxiv.org/abs/1609.09565.

[21] Navid Azizan et al. "Distributed Solution of Large-Scale Linear Systems via Accelerated Projection-Based Consensus". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 6358–6362. DOI: 10.1109/ICASSP.2018.8462630.

[22] Navid Azizan et al. "Distributed Solution of Large-Scale Linear Systems via Accelerated Projection-Based Consensus". In: *IEEE Transactions on Signal Processing* 67.14 (2019), pp. 3806–3817. DOI: 10.1109/TSP.2019.2917855.

[23] Navid Azizan et al. "Improved bounds on the epidemic threshold of exact SIS models on complex networks". In: *2016 55th IEEE Conference on Decision and Control (CDC)*. 2016, pp. 3560–3565. DOI: 10.1109/CDC.2016.7798804.

[24] Navid Azizan et al. “Opportunities for Price Manipulation by Aggregators in Electricity Markets”. In: *SIGMETRICS Performance Evaluation Review* 44.2 (2016), pp. 49–51. ISSN: 0163-5999. DOI: 10.1145/3003977.3003995.

[25] Navid Azizan et al. “Opportunities for Price Manipulation by Aggregators in Electricity Markets”. In: *IEEE Transactions on Smart Grid* 9.6 (2018), pp. 5687–5698. DOI: 10.1109/TSG.2017.2694043.

[26] Navid Azizan et al. “Optimal Pricing in Markets with Non-Convex Costs”. In: *Proceedings of the 2019 ACM Conference on Economics and Computation (EC)*. Phoenix, AZ, USA, 2019, p. 595. ISBN: 978-1-4503-6792-9. DOI: 10.1145/3328526.3329575.

[27] Navid Azizan et al. “Optimal Pricing in Markets with Nonconvex Costs”. In: *Operations Research* 68.2 (2020), pp. 480–496. DOI: 10.1287/opre.2019.1900.

[28] Navid Azizan et al. “Stochastic Mirror Descent on Overparameterized Nonlinear Models: Convergence, Implicit Regularization, and Generalization”. In: *2019 International Conference on Machine Learning (ICML) Generalization Workshop*. 2019.

[29] Norman TJ Bailey et al. *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

[30] Alain Barrat et al. *Dynamical processes on complex networks*. Cambridge University Press, 2008.

[31] Tamer Başar and Pierre Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[32] Paulina Beato and Andreu Mas-Colell. “On marginal cost pricing with given tax-subsidy rules”. In: *Journal of Economic Theory* 37.2 (1985), pp. 356–365.

[33] Amir Beck and Marc Teboulle. “Mirror descent and nonlinear projected subgradient methods for convex optimization”. In: *Operations Research Letters* 31.3 (2003), pp. 167–175.

[34] Omar Ben-Ayed and Charles E Blair. “Computational difficulties of bilevel linear programming”. In: *Operations Research* 38.3 (1990), pp. 556–560.

[35] Suzhi Bi and Ying Jun Zhang. “False-data injection attack to control real-time price in electricity market”. In: *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE. 2013, pp. 772–777.

[36] Suzhi Bi and Ying Jun Zhang. “Using Covert Topological Information for Defense Against Malicious Attacks on DC State Estimation”. In: *Selected Areas in Communications, IEEE Journal on* 32.7 (2014), pp. 1471–1485. ISSN: 0733-8716. DOI: 10.1109/JSAC.2014.2332051.

[37] Ake Björck and Victor Pereyra. “Solution of Vandermonde systems of equations”. In: *Mathematics of Computation* 24.112 (1970), pp. 893–903.

[38] Mette Bjørndal and Kurt Jörnsten. “A Partitioning Method that Generates Interpretable Prices for Integer Programming Problems”. In: *Handbook of Power Systems II* (2010), pp. 337–350.

[39] Mette Bjørndal and Kurt Jörnsten. “Equilibrium prices supported by dual price functions in markets with non-convexities”. In: *European Journal of Operational Research* 190.3 (2008), pp. 768–789.

[40] Elizabeth Bodine-Baron et al. “Minimizing the social cost of an epidemic”. In: *Game Theory for Networks*. Springer, 2012, pp. 594–607.

[41] Severin Borenstein et al. “Market power in California electricity markets”. In: *Utilities Policy* 5.3 (1995), pp. 219–236.

[42] S. Bose et al. “A unifying market power measure for deregulated transmission-constrained electricity markets”. In: *Power Systems, IEEE Transactions on* (2015).

[43] Subhonmesh Bose et al. “The cost of an epidemic over a complex network: A random matrix approach”. In: *arXiv preprint arXiv:1309.2236* (2013).

[44] 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.

[45] Randall Bramley and Ahmed Sameh. “Row projection methods for large nonsymmetric linear systems”. In: *SIAM Journal on Scientific and Statistical Computing* 13.1 (1992), pp. 168–193.

[46] Donald J Brown. “Equilibrium analysis with non-convex technologies”. In: *Handbook of mathematical economics* 4 (1991), pp. 1963–1995.

[47] Kevin Bullis. *Why SolarCity Is Succeeding in a Difficult Solar Industry*. 2012.

[48] James Bushnell et al. “An international comparison of models for measuring market power in electricity”. In: *Energy Modeling Forum Stanford University*. 1999.

[49] California Independent System Operator. *Market power and competitiveness*. `http://www.caiso.com/1c5f/1c5fbe6a1a720.html`. 1998.

[50] California Independent System Operator. *Pricing Enhancements: Revised Straw Proposal*. `https://www.caiso.com/Documents/RevisedStrawProposal\_PricingEnhancements.pdf`. 2014.

[51] Yuan Cao and Quanquan Gu. “A Generalization Theory of Gradient Descent for Learning Over-parameterized Deep ReLU Networks”. In: *arXiv preprint arXiv:1902.01384* (2019).

[52] Judith B Cardell et al. “Market power and strategic interaction in electricity networks”. In: *Resource and energy economics* 19.1 (1997), pp. 109–137.

[53] Claudio Castellano and Romualdo Pastor-Satorras. “Thresholds for epidemic spreading in networks”. In: *Physical review letters* 105.21 (2010), p. 218701.

[54] Eric Cator and Piet Van Mieghem. “Second-order mean-field susceptible-infected-susceptible epidemic threshold”. In: *Physical review E* 85.5 (2012), p. 056111.

[55] Nicolo Cesa-Bianchi et al. “Mirror descent meets fixed share (and feels no regret)”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 980–988.

[56] Meeyoung Cha et al. “A measurement-driven analysis of information propagation in the flickr social network”. In: *Proceedings of the 18th international conference on World wide web*. ACM. 2009, pp. 721–730.

[57] Deepayan Chakrabarti et al. “Epidemic thresholds in real networks”. In: *ACM Transactions on Information and System Security (TISSEC)* 10.4 (2008), p. 1.

[58] Pratik Chaudhari and Stefano Soatto. “Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks”. In: *International Conference on Learning Representations*. 2018.

[59] Yue Chen et al. “Individual risk in mean field control with application to automated demand response”. In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 6425–6432.

[60] Carleton Coffrin et al. “NESTA, the NICTA energy system test case archive”. In: *arXiv:1411.0359* (2014).

[61] Stephan Dempe et al. *Bilevel Programming Problems: Theory, Algorithms and Applications to Energy Networks*. Springer, 2015.

[62] Wei Deng and Wotao Yin. “On the global and linear convergence of the generalized alternating direction method of multipliers”. In: *Journal of Scientific Computing* (2012).

[63] Steven P Dirkse and Michael C Ferris. “The path solver: a nommonotone stabilization scheme for mixed complementarity problems”. In: *Optimization Methods and Software* 5.2 (1995), pp. 123–156.

[64] Moez Draief and Laurent Massouli. *Epidemics and rumours in complex networks*. Cambridge University Press, 2010.

[65] Moez Draief et al. “Thresholds for virus spread on networks”. In: *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*. ACM. 2006, p. 51.

[66] Kimon Drakopoulos et al. “An efficient curing policy for epidemics on graphs”. In: *Network Science and Engineering, IEEE Transactions on* 1.2 (2014), pp. 67–75.

[67] Simon S Du et al. “Gradient descent finds global minima of deep neural networks”. In: *arXiv preprint arXiv:1811.03804* (2018).

[68] John Duchi et al. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[69] Iain S Duff et al. “The augmented block Cimmino distributed method”. In: *SIAM Journal on Scientific Computing* 37.3 (2015), A1248–A1269.

[70] Sanghamitra Dutta et al. “Short-Dot: Computing Large Linear Transforms Distributedly Using Coded Short Dot Products”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 2092–2100.

[71] Electric Reliability Council of Texas. *ERCOT protocols*. `http://www.ercot.com`. 2001.

[72] A Fall et al. “Epidemiological models and Lyapunov functions”. In: *Math. Model. Nat. Phenom* 2.1 (2007), pp. 62–68.

[73] Michael C Ferris et al. “Mathematical programs with equilibrium constraints: Automatic reformulation and solution via constrained optimization”. In: (2002).

[74] Bruce A Francis. *A course in H-infinity control theory*. Berlin; New York: Springer-Verlag, 1987.

[75] Francisco D Galiana et al. “Reconciling social welfare, agent profits, and consumer payments in electricity pools”. In: *IEEE Transactions on Power Systems* 18.2 (2003), pp. 452–459.

[76] Ayalvadi Ganesh et al. “The effect of network topology on the spread of epidemics”. In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Vol. 2. IEEE. 2005, pp. 1455–1466.

[77] Rainer Gemulla et al. “Large-scale matrix factorization with distributed stochastic gradient descent”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, pp. 69–77.

[78] Claudio Gentile. “The robustness of the p-norm algorithms”. In: *Machine Learning* 53.3 (2003), pp. 265–299.

[79] S Gómez et al. “Discrete-time Markov chain approach to contact-based disease spreading in complex networks”. In: *EPL (Europhysics Letters)* 89.3 (2010), p. 38009.

[80] Alex Graves et al. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[81] Paul R Gribik et al. “Market-clearing electricity prices and energy uplift”. In: (2007).

[82] Adam J Grove et al. “General convergence results for linear discriminant updates”. In: *Machine Learning* 43.3 (2001), pp. 173–210.

[83] Roger Guesnerie. “Pareto optimality in non-convex economies”. In: *Econometrica: Journal of the Econometric Society* (1975), pp. 1–29.

[84] Zeynep H Gümüş and Christodoulos A Floudas. “Global optimization of nonlinear bilevel programming problems”. In: *Journal of Global Optimization* 20.1 (2001), pp. 1–31.

[85] Suriya Gunasekar et al. “Characterizing Implicit Bias in Terms of Optimization Geometry”. In: *International Conference on Machine Learning*. 2018, pp. 1827–1836.

[86] Suriya Gunasekar et al. “Implicit Bias of Gradient Descent on Linear Convolutional Networks”. In: *arXiv preprint arXiv:1806.00468* (2018).

[87] Suriya Gunasekar et al. “Implicit Regularization in Matrix Factorization”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6152–6160.

[88] W. Halbawi et al. “Sparse and Balanced Reed–Solomon and Tamo–Barg Codes”. In: *IEEE Transactions on Information Theory* 65.1 (2019), pp. 118–130. DOI: 10.1109/TIT.2018.2873128.

[89] Wael Halbawi et al. “Improving Distributed Gradient Descent Using Reed-Solomon Codes”. In: *2018 IEEE International Symposium on Information Theory (ISIT)*. 2018, pp. 2027–2031. DOI: 10.1109/ISIT.2018.8437467.

[90] W Halbawi et al. “Balanced Reed-Solomon codes”. In: *2016 IEEE International Symposium on Information Theory (ISIT)*. 2016, pp. 935–939. DOI: 10.1109/ISIT.2016.7541436.

[91] W Halbawi et al. “Balanced Reed-Solomon codes for all parameters”. In: *2016 IEEE Information Theory Workshop (ITW)*. 2016, pp. 409–413. DOI: 10.1109/ITW.2016.7606866.

[92] He Hao et al. “Ancillary service for the grid via control of commercial building hvac systems”. In: *American Control Conference (ACC), 2013*. IEEE. 2013, pp. 467–472.

[93] Mor Harchol-Balter. “The Effect of Heavy-Tailed Job Size Distributions on Computer System Design.” In: *Proc. of ASA-IMS Conf. on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics*. 1999.

[94] Mor Harchol-Balter and Allen B Downey. “Exploiting process lifetime distributions for dynamic load balancing”. In: *ACM Transactions on Computer Systems (TOCS)* 15.3 (1997), pp. 253–285.

[95] Babak Hassibi and Thomas Kailath. "Hoo Optimal Training Algorithms and their Relation to Backpropagation". In: *Advances in Neural Information Processing Systems 7*. 1995, pp. 191–198.

[96] Babak Hassibi et al. "Hoo Optimality Criteria for LMS and Backpropagation". In: *Advances in Neural Information Processing Systems 6*. 1994, pp. 351–358.

[97] Babak Hassibi et al. "Hoo optimality of the LMS algorithm". In: *IEEE Transactions on Signal Processing* 44.2 (1996), pp. 267–280.

[98] Babak Hassibi et al. *Indefinite-Quadratic Estimation and Control: A Unified Approach to H2 and H-infinity Theories*. Vol. 16. SIAM, 1999.

[99] Elad Hazan. "Introduction to Online Convex Optimization". In: *Foundations and Trends in Optimization* 2.3-4 (2016), pp. 157–325. ISSN: 2167-3888.

[100] Bingsheng He and Xiaoming Yuan. "On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method". In: *SIAM Journal on Numerical Analysis* (2012).

[101] 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.

[102] Herbert W Hethcote. "The mathematics of infectious diseases". In: *SIAM review* 42.4 (2000), pp. 599–653.

[103] William W Hogan and Brendan J Ring. "On minimum-uplift pricing for electricity markets". In: *Electricity Policy Group* (2003).

[104] Cory Honeyman. *U.S. Residential Solar Economic Outlook 2016-2020*. `http://www.greentechmedia.com/research/report/us-residential-solar-economic-outlook-2016-2020`. Report.

[105] *http://nyssmartgrid.com/innovation-highlights/rev-proceeding*.

[106] *http://www.utilitydive.com/news/ferc-grants-nyiso-request-to-give-behind-the-meter-resources-market-access/419791/*.

[107] Bowen Hua and Ross Baldick. "A convex primal formulation for convex hull pricing". In: *IEEE Transactions on Power Systems* 32.5 (2017), pp. 3814–3823.

[108] David Jacobson. "Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games". In: *IEEE Transactions on Automatic control* 18.2 (1973), pp. 124–131.

[109] Philippe Jacquet et al. "Information propagation speed in mobile and delay tolerant networks". In: *Information Theory, IEEE Transactions on* 56.10 (2010), pp. 5001–5015.

[110] J John. *SolarCity and Tesla: a utility's worst nightmare?* `http://www.greentechmedia.com/articles/read/SolarCitys-Networked-Grid-Ready-Energy-Storage-Fleet`. 2014.

[111] Stefan Kaczmarz. “Angenäherte auflösung von systemen linearer gleichungen”. In: *Bulletin International de l’Academie Polonaise des Sciences et des Lettres* 35 (1937), pp. 355–357.

[112] Kenji Kawaguchi. “Deep learning without poor local minima”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 586–594.

[113] William O Kermack and Anderson G McKendrick. “A contribution to the mathematical theory of epidemics”. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 115. 772. The Royal Society. 1927, pp. 700–721.

[114] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[115] Jyrki Kivinen et al. “The p-norm generalization of the LMS algorithm for adaptive filtering”. In: *IEEE Transactions on Signal Processing* 54.5 (2006), pp. 1782–1793.

[116] B. Kocuk et al. “Inexactness of SDP Relaxation and Valid Inequalities for Optimal Power Flow”. In: *IEEE Transactions on Power Systems* 31.1 (2016), pp. 642–651. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2015.2402640.

[117] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[118] Alex Krizhevsky et al. “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[119] Yann LeCun et al. “Deep learning”. In: *Nature* 521.7553 (2015), p. 436.

[120] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[121] Yann LeCun et al. *The MNIST database of handwritten digits*. 1998.

[122] Jason D Lee et al. “Gradient descent only converges to minimizers”. In: *Conference on Learning Theory*. 2016, pp. 1246–1257.

[123] Kangwook Lee et al. “Speeding up distributed machine learning using codes”. In: *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 1143–1147.

[124] Will Leland and Teunis J Ott. *Load-balancing heuristics and process behavior*. Vol. 14. 1. ACM, 1986.

[125] Laurent Lessard et al. “Analysis and design of optimization algorithms via integral quadratic constraints”. In: *SIAM Journal on Optimization* 26.1 (2016), pp. 57–95.

[126] David Asher Levin et al. *Markov chains and mixing times*. American Mathematical Soc., 2009.

[127] Chun-Hsien Li et al. “Analysis of epidemic spreading of an SIRS model in complex heterogeneous networks”. In: *Communications in Nonlinear Science and Numerical Simulation* 19.4 (2014), pp. 1042–1054.

[128] Songze Li et al. “Fundamental tradeoff between computation and communication in distributed computing”. In: *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 1814–1818.

[129] Yuanzhi Li and Yingyu Liang. “Learning overparameterized neural networks via stochastic gradient descent on structured data”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 8157–8166.

[130] Guanfeng Liang and Ulas C Kozat. “TOFEC: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes”. In: *INFOCOM, 2014 Proceedings IEEE*. IEEE. 2014, pp. 826–834.

[131] George Liberopoulos and Panagiotis Andrianesis. “Critical review of pricing schemes in markets with non-convex costs”. In: *Operations Research* 64.1 (2016), pp. 17–31.

[132] Yashen Lin et al. “Experimental evaluation of frequency regulation from commercial building HVAC systems”. In: *Smart Grid, IEEE Transactions on* 6.2 (2015), pp. 776–783.

[133] Nicole Litvak. *U.S. Commercial Solar Landscape 2016-2020*. `http://www.greentechmedia.com/research/report/us-commercial-solar-landscape-2016-2020`. Report.

[134] Ji Liu et al. “An asynchronous distributed algorithm for solving a linear algebraic equation”. In: *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE. 2013, pp. 5409–5414.

[135] K Liu. *CIFAR10 with PyTorch*. `https://github.com/kuangliu/pytorch-cifar`.

[136] Yao Liu et al. “False data injection attacks against state estimation in electric power grids”. In: *ACM Transactions on Information and System Security (TISSEC)* 14.1 (2011), p. 13.

[137] Cong Ma et al. “Implicit Regularization in Nonconvex Statistical Estimation: Gradient Descent Converges Linearly for Phase Retrieval and Matrix Completion”. In: *International Conference on Machine Learning*. 2018, pp. 3351–3360.

[138] Siyuan Ma et al. “The Power of Interpolation: Understanding the Effectiveness of SGD in Modern Over-parametrized Learning”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, 2018, pp. 3325–3334.

[139] Layla Majzoubi and Farshad Lahouti. “Analysis of distributed ADMM algorithm for consensus optimization in presence of error”. In: *Proceedings of the 2016 IEEE Confs. Audio, Speech and Signal Processing (ICASSP)*. 2016.

[140] Yuri Makarov et al. “INCORPORATION OF WIND POWER RESOURCES INTO THE CALIFORNIA ENERGY MARKET”. In: ().

[141] *Matrix Market*. `http://math.nist.gov/MatrixMarket/`. Accessed: May 2017.

[142] Poorya Mianjy et al. “On the Implicit Bias of Dropout”. In: *International Conference on Machine Learning*. 2018, pp. 3537–3545.

[143] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), p. 529.

[144] João FC Mota et al. “D-ADMM: A communication-efficient distributed algorithm for separable optimization”. In: *IEEE Transactions on Signal Processing* 61.10 (2013), pp. 2718–2723.

[145] Alexis L Motto and Francisco D Galiana. “Equilibrium of auction markets with unit commitment: The need for augmented pricing”. In: *IEEE Transactions on Power Systems* 17.3 (2002), pp. 798–805.

[146] Shaoshuai Mou et al. “A distributed algorithm for solving a linear algebraic equation”. In: *IEEE Transactions on Automatic Control* 60.11 (2015), pp. 2863–2878.

[147] Angelia Nedic and Soomin Lee. “On stochastic subgradient mirror-descent algorithm with weighted averaging”. In: *SIAM Journal on Optimization* 24.1 (2014), pp. 84–107.

[148] Matias Negrete-Pincetic and Sean Meyn. “Markets for differentiated electric power products in a Smart Grid environment”. In: *Power and Energy Society General Meeting, 2012 IEEE*. IEEE. 2012, pp. 1–7.

[149] Arkadii Nemirovski and David Borisovich Yudin. “Problem complexity and method efficiency in optimization.” In: (1983).

[150] Arkadi Nemirovski et al. “Robust stochastic approximation approach to stochastic programming”. In: *SIAM Journal on optimization* 19.4 (2009), pp. 1574–1609.

[151] Yurii Nesterov. “A method of solving a convex programming problem with convergence rate $O(1/k^2)$”. In: *Soviet Mathematics Doklady*. Vol. 27. 2. 1983, pp. 372–376.

[152] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2013.

[153] Yurii Nesterov. “Primal-dual subgradient methods for convex problems”. In: *Mathematical programming* 120.1 (2009), pp. 221–259.

[154] Behnam Neyshabur et al. “Geometry of optimization and implicit regularization in deep learning”. In: *arXiv preprint arXiv:1705.03071* (2017).

[155] Cameron Nowzari et al. “Analysis and Control of Epidemics: A Survey of Spreading Processes on Complex Networks”. In: *Control Systems, IEEE* 36.1 (2016), pp. 26–46.

[156] Cameron Nowzari et al. “Analysis and Control of Epidemics: A survey of spreading processes on complex networks”. In: *arXiv preprint arXiv:1505.00768* (2015).

[157] Cameron Nowzari et al. “Stability analysis of generalized epidemic models over directed networks”. In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 6197–6202.

[158] Elli Ntakou and Michael Caramanis. “Distribution network electricity market clearing: Parallelized PMP algorithms with minimal coordination”. In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 1687–1694.

[159] Richard P O’Neill et al. “Efficient market-clearing prices in markets with nonconvexities”. In: *European journal of operational research* 164.1 (2005), pp. 269–285.

[160] Masaki Ogura and Victor M Preciado. “Second-Order Moment-Closure for Tighter Epidemic Thresholds”. In: *arXiv preprint arXiv:1706.08602* (2017).

[161] Shmuel S Oren. “Economic inefficiency of passive transmission rights in congested electricity systems with competitive generation”. In: *The Energy Journal* (1997), pp. 63–83.

[162] Andrew L Ott. “Experience with PJM market operation, system design, and implementation”. In: *Power Systems, IEEE Transactions on* 18.2 (2003), pp. 528–534.

[163] Samet Oymak and Mahdi Soltanolkotabi. “Overparameterized Nonlinear Learning: Gradient Descent Takes the Shortest Path?” In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019.

[164] Xinghao Pan et al. “Revisiting Distributed Synchronous SGD”. In: *arXiv preprint arXiv:1702.05800* (2017).

[165] Romualdo Pastor-Satorras et al. “Epidemic processes in complex networks”. In: *Reviews of modern physics* 87.3 (2015), p. 925.

[166] David B Patton et al. “2013 assessment of the electricity markets in new england”. In: *Potomac Economics* (2014).

[167] Mathew Penrose. *Random geometric graphs*. Vol. 5. Oxford University Press Oxford, 2003.

[168] Joseph E Phelps et al. “Viral marketing or electronic word-of-mouth advertising: Examining consumer responses and motivations to pass along email”. In: *Journal of advertising research* 44.04 (2004), pp. 333–348.

[169] 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.

[170] B Aditya Prakash et al. “Threshold conditions for arbitrary cascade models on arbitrary networks”. In: *Knowledge and information systems* 33.3 (2012), pp. 549–575.

[171] Maxim Raginsky and Jake Bouvrie. “Continuous-time stochastic mirror descent on a network: Variance reduction, consensus, convergence”. In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE. 2012, pp. 6793–6800.

[172] Netanel Raviv et al. “Gradient Coding from Cyclic MDS Codes and Expander Graphs”. In: *arXiv preprint arXiv:1707.03858* (2017).

[173] Benjamin Recht et al. “Hogwild: A lock-free approach to parallelizing stochastic gradient descent”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 693–701.

[174] Irving Reed and Gus Solomon. “Polynomial codes over certain finite fields”. In: *Journal of the Society for Industrial & Applied Mathematics* (1960).

[175] Matthew Richardson and Pedro Domingos. “Mining knowledge-sharing sites for viral marketing”. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2002, pp. 61–70.

[176] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* (1951), pp. 400–407.

[177] Carlos Ruiz et al. “Pricing non-convexities in an electricity pool”. In: *IEEE Transactions on Power Systems* 27.3 (2012), pp. 1334–1342.

[178] Herbert E Scarf. “Mathematical programming and economic theory”. In: *Operations Research* 38.3 (1990), pp. 377–385.

[179] Herbert E Scarf. “The allocation of resources in the presence of indivisibilities”. In: *The Journal of Economic Perspectives* 8.4 (1994), pp. 111–128.

[180] David T Scheffman and Pablo T Spiller. “Geographic market definition under the US Department of Justice Merger Guidelines”. In: *Journal of Law and Economics* (1987), pp. 123–147.

[181] Dane A Schiro et al. “Convex Hull Pricing in Electricity Markets: Formulation, Analysis, and Implementation Challenges”. In: *IEEE Transactions on Power Systems* 31.5 (2016), pp. 4068–4075.

[182] Richard Schmalensee and Bennett W Golub. "Estimating effective concentration in deregulated wholesale electricity markets". In: *The RAND Journal of Economics* (1984), pp. 12–26.

[183] Shai Shalev-Shwartz. "Online Learning and Online Convex Optimization". In: *Foundations and Trends in Machine Learning* 4.2 (2012), pp. 107–194. ISSN: 1935-8237.

[184] A Sheffrin and Jing Chen. "Predicting market power in wholesale electricity markets". In: *Proc. of the Western Conference of the Advances in Regulation and Competition, South Lake Tahoe*. 2002.

[185] Wei Shi et al. "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization." In: *IEEE Trans. Signal Processing* 62.7 (2014), pp. 1750–1761.

[186] *Shift factor: methodology and example*. `http://www.caiso.com/docs/2004/02/13/200402131609438684.pdf`. 2004.

[187] Zhisheng Shuai and Pauline van den Driessche. "Global stability of infectious disease models using Lyapunov functions". In: *SIAM Journal on Applied Mathematics* 73.4 (2013), pp. 1513–1532.

[188] Ravid Shwartz-Ziv and Naftali Tishby. "Opening the black box of deep neural networks via information". In: *arXiv preprint arXiv:1703.00810* (2017).

[189] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[190] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[191] Fridrich Sloboda. "A projection method of the Cimmino type for linear algebraic systems". In: *Parallel computing* 17.4-5 (1991), pp. 435–442.

[192] *Solar Energy Industries Association (SEIA)*. `http://www.seia.org/state-solar-policy`. Accessed: 2016-05-26.

[193] Mahdi Soltanolkotabi et al. "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks". In: *arXiv preprint arXiv:1707.04926* (2017).

[194] Daniel Soudry et al. "The implicit bias of gradient descent on separable data". In: *arXiv preprint arXiv:1710.10345* (2017).

[195] Jason L Speyer et al. "Optimal stochastic estimation with exponential cost criteria". In: *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*. IEEE. 1992, pp. 2293–2299.

[196] Jason Speyer et al. "Optimization of stochastic linear systems with additive measurement and process noise using exponential performance criteria". In: *IEEE Transactions on Automatic Control* 19.4 (1974), pp. 358–366.

[197] Rashish Tandon et al. “Gradient Coding”. In: *NIPS*. 2016, pp. 1–12. arXiv: 1612.03301. URL: http://arxiv.org/abs/1612.03301.

[198] Rashish Tandon et al. “Gradient Coding: Avoiding Stragglers in Distributed Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 3368–3376. URL: http://proceedings.mlr.press/v70/tandon17a.html.

[199] Tijmen Tieleman and Geoffrey Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[200] John Nikolas Tsitsiklis. *Problems in decentralized decision making and computation.* Tech. rep. Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984.

[201] Ralph Turvey. “Marginal cost”. In: *The Economic Journal* 79.314 (1969), pp. 282–299.

[202] Paul Twomey and Karsten Neuhoff. “Wind power and market power in competitive markets”. In: *Energy Policy* 38.7 (2010), pp. 3198–3210.

[203] Paul Twomey et al. *A Review of the Monitoring of Market Power: The Possible Roles of TSOs in Monitoring for Market Power Issues in Congested Transmission Systems*. Tech. Report. 2015.

[204] Piet Van Mieghem. “Decay towards the overall-healthy state in SIS epidemics on networks”. In: *arXiv preprint arXiv:1310.3980* (2013).

[205] Piet Van Mieghemy et al. “An upper bound for the epidemic threshold in exact Markovian SIR and SIS epidemics on networks”. In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 6228–6233.

[206] Kerstin Vännman. “Estimators based on order statistics from a Pareto distribution”. In: *Journal of the American Statistical Association* 71.355 (1976), pp. 704–708.

[207] Xuan Wang et al. “A Distributed Algorithm for Least Square Solutions of Linear Equations”. In: *arXiv preprint arXiv:1709.10157* (2017).

[208] Yang Wang et al. “Epidemic spreading in real networks: An eigenvalue viewpoint”. In: *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on*. IEEE. 2003, pp. 25–34.

[209] Eric Wesoff. *Earnings From SunPower, Tesla, Enphase, Plus New Funding for Sunnova, SolarCity, 1366, Siva, Nexeon*. http://www.greentechmedia.com/articles/read/Earnings-from-SunPower-Tesla-Enphase-and-New-Funding-for-Sunnova-SolarCi. 2016.

[210] Peter Whittle. *Risk-sensitive optimal control*. John Wiley & Son Ltd, 1990.

[211] Ashia C Wilson et al. “The marginal value of adaptive gradient methods in machine learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4151–4161.

[212] Laurence A Wolsey. “Integer programming duality: Price functions and sensitivity analysis”. In: *Mathematical Programming* 20.1 (1981), pp. 173–195.

[213] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[214] Lin Xiao. “Dual averaging methods for regularized stochastic learning and online optimization”. In: *Journal of Machine Learning Research* 11.Oct (2010), pp. 2543–2596.

[215] Lin Xiao and Stephen Boyd. “Fast linear iterations for distributed averaging”. In: *Systems & Control Letters* 53.1 (2004), pp. 65–78.

[216] Le Xie et al. “False data injection attacks in electricity markets”. In: *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE. 2010, pp. 226–231.

[217] Le Xie et al. “Integrity data attacks in power market operations”. In: *Smart Grid, IEEE Transactions on* 2.4 (2011), pp. 659–666.

[218] Lin Xu and Ross Baldick. “Transmission-constrained residual demand derivative in electricity markets”. In: *Power Systems, IEEE Transactions on* 22.4 (2007), pp. 1563–1573.

[219] Lin Xu and Yixin Yu. “Transmission constrained linear supply function equilibrium in power markets: method and example”. In: *Power System Technology, 2002. Proceedings. PowerCon 2002. International Conference on*. Vol. 3. IEEE. 2002, pp. 1349–1354.

[220] Qian Yu et al. “Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication”. In: *arXiv preprint arXiv:1705.10464* (2017).

[221] Yang Yu et al. “Do wind power producers have market power and exercise it?” In: *PES General Meeting| Conference & Exposition, 2014 IEEE*. IEEE. 2014, pp. 1–5.

[222] Kun Yuan et al. “On the convergence of decentralized gradient descent”. In: *SIAM Journal on Optimization* 26.3 (2016), pp. 1835–1854.

[223] Baosen Zhang et al. “An optimal and distributed method for voltage regulation in power distribution systems”. In: *Power Systems, IEEE Transactions on* 30.4 (2015), pp. 1714–1726.

[224] Chiyuan Zhang et al. “Understanding deep learning requires rethinking generalization”. In: *arXiv preprint arXiv:1611.03530* (2016).

[225] Ruiliang Zhang and James T Kwok. “Asynchronous Distributed ADMM for Consensus Optimization.” In: *ICML*. 2014, pp. 1701–1709.

[226] Tongxin Zheng and Eugene Litvinov. “Ex post pricing in the co-optimized energy and reserve market”. In: *IEEE Trans. on Power Sys.* 21.4 (2006), pp. 1528–1538.

[227] Zhengyuan Zhou et al. “Stochastic Mirror Descent in Variationally Coherent Optimization Problems”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 7043–7052.

[228] Martin Zinkevich et al. “Parallelized stochastic gradient descent”. In: *Advances in neural information processing systems*. 2010, pp. 2595–2603.