TOPICS IN SPARSE MULTIVARIATE STATISTICS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF STATISTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Rahul Mazumder

August 2012

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Trevor Hastie) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Jerome Friedman)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Robert Tibshirani)

Approved for the University Committee on Graduate Studies

Abstract

In this thesis, we visit three topics in modern sparse multivariate analysis that has received and still continues to receive significant interest in applied statistics in recent years:

- Sparse high dimensional regression
  (This work appears in our paper Mazumder et al. (2010).)

- Low-rank matrix completion & collaborative filtering
  (This work appears in our paper Mazumder et al. (2011).)

- Sparse undirected gaussian graphical models
  (This work appears in our papers Mazumder & Hastie (2012) and Mazumder & Hastie (2011).)

A main challenge in high dimensional multivariate analysis is in developing scalable and efficient algorithms for large scale problems that naturally arise in scientific and industrial applications. This thesis explores the computational challenges that arise in these problems. We develop and analyze statistically motivated algorithms for various models arising in the aforementioned areas. This enhances our understanding of algorithms, sheds novel insights on the statistical problems and leads to computational strategies that appear to outperform state of the art. A salient feature of this work lies in the exchange of ideas and machinery across the fields of statistics, machine learning, (convex) optimization and numerical linear algebra.

Acknowledgements

This thesis would not have been possible without the support of people who stood by me during my years at Stanford. I would like to convey my sincere thanks to all of them.

Firstly, I would like to thank my adviser, Trevor Hastie, and all the other thesis committee-members: Emmanuel Candes, Jerome Friedman, Michael Saunders and Robert Tibshirani for being a part of my committee.

I would like to express my appreciation and deep gratitude to Trevor for being a great adviser, a great teacher and a wonderful person to work with. I have learned many good things from him both on the academic and personal front. He has been there to share ideas and at the same time encourage me to pursue my own. He has been extremely enthusiastic and supportive throughout my PhD years. My journey in applied statistics and machine learning would not have been possible without him. A big thanks to the Hasties, specially Lynda, for the numerous delicious dinners and being great hosts.

I would like to convey my humble thanks to Jerome Friedman, for being a great mentor. Jerry's tremendous intellectual energy, enthusiasm, depth of knowledge and deep thinking have inspired me a lot. I immensely benefited from discussions with him on various topics related to this thesis — I am indebted to him for his time and patience. I would like to take this opportunity to express my sincere thanks to Emmanuel Candes, for his advice and discussions related to various academic topics, especially Convex Optimization and Matrix Completion. Emmanuel's enthusiasm, passion and positive energy will have a lasting impression on me. I greatly value his feedback and suggestions during my presentations at his group meetings. I would like to convey my regards to Jonathan Taylor. He has always been there to discuss ideas and talk about different academic topics related or unrelated to this thesis. Thanks also to Robert Tibshirani, for being a co-author in the matrix completion work. I greatly acknowledge his support and feedback on various occasions during the course of the last five years. A big thanks to Naras, for being supportive and a great friend. Finally, I would like to express my appreciation to the all the Stanford faculty members whom I came across at some point or the other, during my stint as a graduate student. A big thanks goes to Deepak Agarwal, my mentor at Yahoo! Research (during my summer internships in 2009 and 2010) for being a great collaborator and friend.

My appreciation goes out to all the members of the Hastie-Tibshirani-Taylor research group; Emmanuel's research group and the other students of the Statistics department for creating an intellectually stimulating environment.

A big thanks to my friends and colleagues at Stanford, specially: Anirban, Arian, Bhaswar, Brad, Camilor, Dennis, Ewout, Gourab, Jason, Jeremy, Kshitij-da, Leo, Shilin, Sumit, Victor, Will, Zhen — without whom my stay here would have been nowhere as enjoyable. A warmly acknowledge the support of some of my close friends from ISI, specially GP and Saha for being there beside me through my ups and downs. I would also like to reach out to Debapriya Sengupta from ISI and express my gratitude for encouraging me to embark on this adventure.

I would like to thank my parents and brother for their endless love and support and letting me pursue a PhD at Stanford, far away from home. My parents have always stood by me and encouraged me to pursue my dreams — I dedicate this thesis to them for their encouragement, sacrifice and unconditional support. A big thanks Lisa, for her love and support; and for being a wonderful person to be with.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A principal task of modern statistical learning is to make meaningful inference based on available data which comes in diverse forms from genomic, internet, finance, environmental, marketing (among other) applications. In the 'high dimensional' setting a lot of exciting research is being conducted in learning sparse, parsimonious and interpretable models via the ℓ_1 penalization (aka LASSO) framework (Tibshirani 2011, for example). A large part of this is motivated by high-impact real-life applications. The advent of new, complex and large-scale data-structures from different applications in science and industry requires extending the scope of existing methods to structured sparse regression, matrix regularization and covariance estimation. An important and challenging research direction in this context is in developing statistical models for these tasks, building efficient algorithms for them and exploring the trade-offs between algorithmic scalability and statistical inference. This thesis addresses this important theme. What is particularly exciting in this effort is the cross-fertilization of concepts and tools across the disciplines of statistics and machine learning, convex optimization (and analysis) and numerical linear algebra.

This thesis covers three topics, based on a series of papers by us; and can be classified under under the following broad divisions:

Part I: *Non-Convex Penalized Sparse Regression*

This chapter develops *df* calibrated path-approaches to non-convex penalized regression, significantly improving upon existing algorithms for this task.

Part I of the thesis is based on joint work with Trevor Hastie and Jerome Friedman and appears in our paper Mazumder et al. (2011);

Part II: Matrix Completion and Collaborative Filtering

This part studies (convex) nuclear-norm relaxations of (non-convex) low-rank constraints and exploring the connections with more traditional low-rank matrix factorization based methods. Certain non-convex variants of the rank penalty are also covered.

Part II of the thesis is based on joint work with Trevor Hastie and Robert Tibshirani and appears in our paper Mazumder et al. (2010);

Part III: Covariance Selection & Graphical Lasso : New Insights & Alternatives

This part of the thesis is further sub-divided into Chapters‘4 and 5:

- *Exact Covariance Thresholding for* GLASSO :

  This is about a previously unknown and interesting property of Graphical Lasso aka GLASSO which establishes the equivalence between the connected components of the sparse precision graph and the same of a thresholded version of the sample covariance matrix. This extends the scope of applications to problems with number of nodes ranging from tens of thousands to possibly a million (current technology has trouble in scaling beyond 1-2 K nodes).

  This part of the thesis is based on joint work with Trevor Hastie, and appears in our paper Mazumder & Hastie (2012).

- GLASSO *New Insights and Alternatives*:

  This chapter highlights some of the properties of the GLASSO algorithm (Friedman, Hastie & Tibshirani 2007) that appear mysterious and provides explanations to them. In the process of our analysis, we propose new (and superior) algorithm(s) on the primal: P-GLASSO and DP-GLASSO and justify their usefulness.

This part is based on joint work with Trevor Hastie, and can be found in our article Mazumder & Hastie (2011).

Part I

Non-Convex Penalized Sparse Regression

Chapter 2

SparseNet: Non-Convex Coordinate Descent

Abstract: In this chapter we address the problem of sparse selection in linear models. A number of non-convex penalties have been proposed in the literature for this purpose, along with a variety of convex-relaxation algorithms for finding good solutions. In this chapter we pursue a coordinate-descent approach for optimization, and study its convergence properties. We characterize the properties of penalties suitable for this approach, study their corresponding threshold functions, and describe a *df*-standardizing reparametrization that assists our pathwise algorithm. The MC+ penalty (Zhang 2010) is ideally suited to this task, and we use it to demonstrate the performance of our algorithm.

The content of this chapter is based on joint work with Jerome Friedman and Trevor Hastie; and the work appears in our paper Mazumder et al. (2011).

2.1 Introduction

Consider the usual linear regression set-up

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.1}$$

with n observations and p features $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$, response $\mathbf{y}$, coefficient vector $\boldsymbol{\beta}$ and (stochastic) error $\boldsymbol{\epsilon}$. In many modern statistical applications with $p \gg n$, the true $\boldsymbol{\beta}$ vector is often sparse, with many redundant predictors having coefficient zero. We would like to identify the useful predictors and also obtain good estimates of their coefficients. Identifying a set of relevant features from a list of many thousand is in general combinatorially hard and statistically troublesome. In this context, convex relaxation techniques such as the LASSO (Tibshirani 1996, Chen & Donoho 1994) have been effectively used for simultaneously producing accurate and parsimonious models. The LASSO solves

$$\min_{\boldsymbol{\beta}} \quad \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1. \tag{2.2}$$

The ℓ_1 penalty shrinks coefficients towards zero, and can also set many coefficients to be exactly zero. In the context of variable selection, the LASSO is often thought of as a convex surrogate for best-subset selection:

$$\min_{\boldsymbol{\beta}} \quad \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_0. \tag{2.3}$$

The ℓ_0 penalty $\|\boldsymbol{\beta}\|_0 = \sum_{i=1}^{p} \mathbf{I}(|\beta_i| > 0)$ penalizes the number of non-zero coefficients in the model.

The LASSO enjoys attractive statistical properties (Zhao & Yu 2006, Donoho 2006, Knight & Fu 2000, Meinshausen & Bühlmann 2006). Under certain regularity conditions on $\mathbf{X}$, it produces models with good prediction accuracy when the underlying model is reasonably sparse. Zhao & Yu (2006) established that the LASSO is model selection consistent: $\Pr(\hat{\mathcal{A}} = \mathcal{A}^0) \to 1$, where $\mathcal{A}^0$ corresponds to the set of nonzero coefficients (active) in the true model and $\hat{\mathcal{A}}$ those recovered by the LASSO. Typical assumptions limit the pairwise correlations between the variables.

However, when these regularity conditions are violated, the LASSO can be sub-optimal in model selection (Zhang 2010, Zhang & Huang 2008, Friedman 2008, Zou & Li 2008, Zou 2006). Since the LASSO both shrinks and selects, it often selects a model which is overly dense in its effort to relax the penalty on the relevant coefficients. Typically in such situations greedier methods like subset regression and the non-convex methods we discuss here achieve sparser models than the LASSO for the same or better prediction accuracy, and enjoy superior variable-selection properties.

There are computationally attractive algorithms for the LASSO. The piecewise-linear LASSO coefficient paths can be computed efficiently via the LARS (homotopy) algorithm (Efron et al. 2004, Osborne et al. 2000). Coordinate-wise optimization algorithms (Friedman et al. 2009) appear to be the fastest for computing the regularization paths for a variety of loss functions, and scale well. One-at-a time coordinate-wise methods for the LASSO make repeated use of the univariate *soft-thresholding* operator

$$\begin{aligned} \mathrm{S}(\tilde{\beta}, \lambda) &= \arg\min_{\beta} \left\{ \tfrac{1}{2}(\beta - \tilde{\beta})^2 + \lambda|\beta| \right\} \\ &= \operatorname{sgn}(\tilde{\beta})(|\tilde{\beta}| - \lambda)_+. \end{aligned} \tag{2.4}$$

In solving (2.2), the one-at-a-time coordinate-wise updates are given by

$$\tilde{\beta}_j = \mathrm{S}\left(\sum_{i=1}^{n} (y_i - \tilde{y}_i^j) x_{ij}, \lambda \right), \quad j = 1, \ldots, p \tag{2.5}$$

where $\tilde{y}_i^j = \sum_{k \neq j} x_{ik} \tilde{\beta}_k$ (assuming each $\mathbf{x}_j$ is standardized to have mean zero and unit ℓ_2 norm). Starting with an initial guess for $\tilde{\boldsymbol{\beta}}$ (typically a solution at the previous value for λ), we cyclically update the parameters using (2.5) until convergence.

The LASSO can fail as a variable selector. In order to get the full effect of a relevant variable, we have to relax the penalty, which lets in other redundant but possibly correlated features. This is in contrast to best-subset regression; once a strong variable is included and fully fit, it drains the effect of its correlated surrogates.

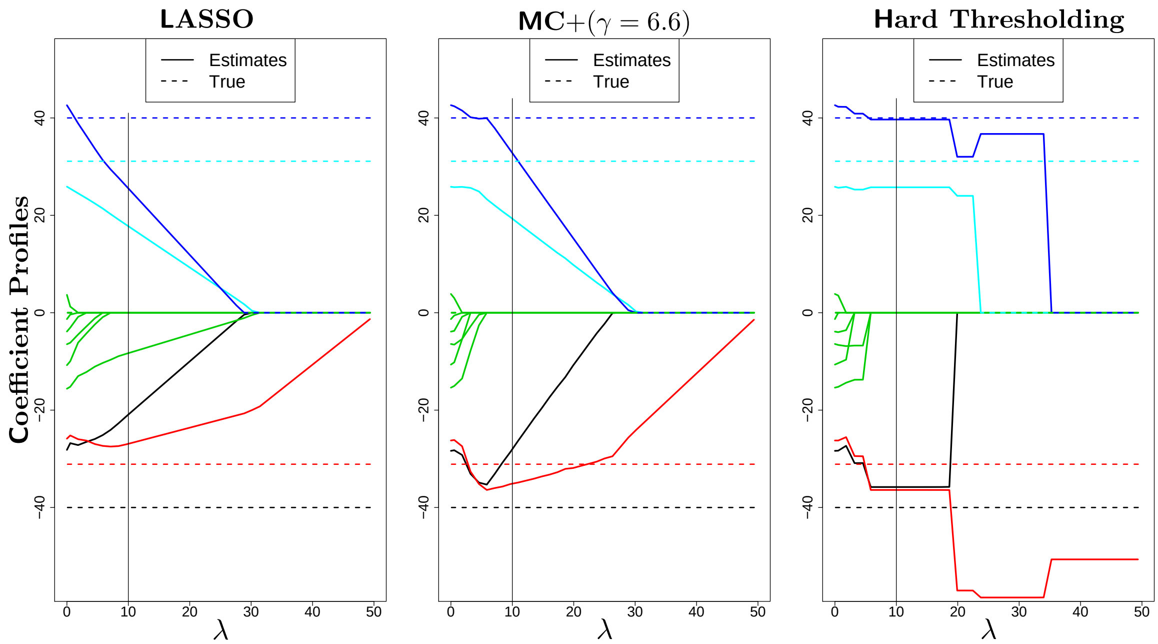

Figure 2.1: Regularization path for LASSO (left), MC+ (non-convex) penalized least-squares for $\gamma = 6.6$ (centre) and $\gamma = 1+$ (right), corresponding to the hard-thresholding operator (best subset). The true coefficients are shown as horizontal dotted lines. Here LASSO is sub-optimal for model selection, as it can never recover the true model. The other two penalized criteria are able to select the correct model (vertical lines), with the middle one having smoother coefficient profiles than best subset on the right.

As an illustration, Figure 2.1 shows the regularization path of the LASSO coefficients for a situation where it is sub-optimal for model selection.[1] This motivates going beyond the ℓ_1 regime to more aggressive *non-convex* penalties (see the left-hand plots for each of the four penalty families in Figure 2.2), bridging the gap between ℓ_1 and ℓ_0 (Fan & Li 2001, Zou 2006, Zou & Li 2008, Friedman 2008, Zhang 2010). Similar to (2.2), we minimize

$$Q(\boldsymbol{\beta}) = \quad \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{i=1}^{p} P(|\beta_i|; \lambda; \gamma), \tag{2.6}$$

where $P(|\beta|; \lambda; \gamma)$ defines a family of penalty functions concave in $|\beta|$, and λ and γ control

[1]The simulation setup is defined in Section 2.7. Here $n = 40$, $p = 10$, SNR$= 3$, $\boldsymbol{\beta} = (-40, -31, \mathbf{0}_{1\times 6}, 31, 40)$ and $X \sim \text{MVN}(0, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \text{diag}[\boldsymbol{\Sigma}(0.65; 5), \boldsymbol{\Sigma}(0; 5)]$.

the degrees of regularization and concavity of the penalty respectively.

The main challenge here is in the minimization of the possibly non-convex objective $Q(\boldsymbol{\beta})$. As one moves down the continuum of penalties from ℓ_1 to ℓ_0, the optimization potentially becomes combinatorially hard.[2]

Our contributions in this chapter are as follows.

1. We propose a coordinate-wise optimization algorithm *SparseNet* for finding minima of $Q(\boldsymbol{\beta})$. Our algorithm cycles through both λ and γ, producing solution surfaces $\hat{\boldsymbol{\beta}}_{\lambda,\gamma}$ for all families simultaneously. For each value of λ, we start at the LASSO solution, and then update the solutions via coordinate descent as γ changes, moving us towards best-subset regression.

2. We study the generalized univariate thresholding functions $S_\gamma(\tilde{\beta}, \lambda)$ that arise from different non-convex penalties (2.6), and map out a set of properties that make them more suitable for coordinate descent. In particular, we seek continuity (in $\tilde{\beta}$) of these threshold functions for both λ and γ.

3. We prove *convergence* of coordinate descent for a useful subclass of non-convex penalties, generalizing the results of Tseng & Yun (2009) to nonconvex problems. Our results go beyond those of Tseng (2001) and Zou & Li (2008); they study stationarity properties of *limit points*, and not convergence of the sequence produced by the algorithms.

4. We propose a re-parametrization of the penalty families that makes them even more suitable for coordinate-descent. Our re-parametrization constrains the coordinate-wise *effective degrees of freedom* at any value of λ to be constant as γ varies. This in turn allows for a natural transition across neighboring solutions as as we move

[2] the optimization problems become non-convex when the non-convexity of the penalty is no longer dominated by the convexity of the squared error loss.

through values of γ from the convex LASSO towards best-subset selection, with the size of the active set decreasing along the way.

5. We compare our algorithm to the state of the art for this class of problems, and show how our approaches lead to improvements.

Note that this chapter is about an algorithm for solving a non-convex optimization problem. What we produce is a good estimate for the solution surfaces. We do not go into methods for selecting the tuning paramaters, nor the properties of the resulting estimators.

The chapter is organized as follows. In Section 2.2 we study four families of non-convex penalties, and their induced thresholding operators. We study their properties, particularly from the point of view of coordinate descent. We propose a degree of freedom (*df*) calibration, and lay out a list of desirable properties of penalties for our purposes. In Section 2.3 we describe our *SparseNet* algorithm for finding a surface of solutions for all values of the tuning parameters. In Section 2.5 we illustrate and implement our approach using the MC+ penalty (Zhang 2010) or the firm shrinkage threshold operator (Gao & Bruce 1997). In Section 2.6 we study the convergence properties of our *SparseNet* algorithm. Section 2.7 presents simulations under a variety of conditions to demonstrate the performance of *SparseNet*. Section 2.8 investigates other approaches, and makes comparisons with *SparseNet* and multi-stage Local Linear Approximation (MLLA/LLA) (Zou & Li 2008, Zhang 2009, Candes et al. 2008). The proofs of lemmas and theorems are gathered in the appendix.

2.2 Generalized Thresholding Operators

In this section we examine the one-dimensional optimization problems that arise in coordinate descent minimization of $Q(\boldsymbol{\beta})$. With squared-error loss, this reduces to minimizing

$$Q^{(1)}(\beta) = \tfrac{1}{2}(\beta - \tilde{\beta})^2 + \lambda P(|\beta|, \lambda, \gamma). \tag{2.7}$$

We study (2.7) for different non-convex penalties, as well as the associated *generalized threshold operator*

$$\mathrm{S}_\gamma(\tilde{\beta}, \lambda) = \arg\min_\beta Q^{(1)}(\beta). \tag{2.8}$$

As γ varies, this generates a family of threshold operators $\mathrm{S}_\gamma(\cdot, \lambda) : \Re \to \Re$. The soft-threshold operator (2.4) of the LASSO is a member of this family. The hard-thresholding operator (2.9) can also be represented in this form

$$\begin{aligned} \mathrm{H}(\tilde{\beta}, \lambda) &= \arg\min_\beta \left\{ \tfrac{1}{2}(\beta - \tilde{\beta})^2 + \lambda\, \mathbf{I}(|\beta| > 0) \right\} \\ &= \tilde{\beta}\, \mathbf{I}\,(|\tilde{\beta}| \geq \lambda). \end{aligned} \tag{2.9}$$

Our interest in thresholding operators arose from the work of She (2009), who also uses them in the context of sparse variable selection, and studies their properties for this purpose. Our approaches differ, however, in that our implementation uses coordinate descent, and exploits the structure of the problem in this context.

For a better understanding of non-convex penalties and the associated threshold operators, it is helpful to look at some examples. For each penalty family (a)–(d), there is a pair of plots in Figure 2.2; the left plot is the penalty function, the right plot the induced thresholding operator.

(a) The ℓ_γ penalty given by $\lambda P(t; \lambda; \gamma) = \lambda |t|^\gamma$ for $\gamma \in [0, 1]$, also referred to as the bridge or power family (Frank & Friedman 1993, Friedman 2008).

(b) The log-penalty is a generalization of the elastic net family (Friedman 2008) to cover the non-convex penalties from LASSO down to best subset.

$$\lambda P(t; \lambda; \gamma) = \frac{\lambda}{\log(\gamma + 1)} \log(\gamma |t| + 1), \gamma > 0, \tag{2.10}$$

where for each value of λ we get the entire continuum of penalties from ℓ_1 ($\gamma \to 0+$)

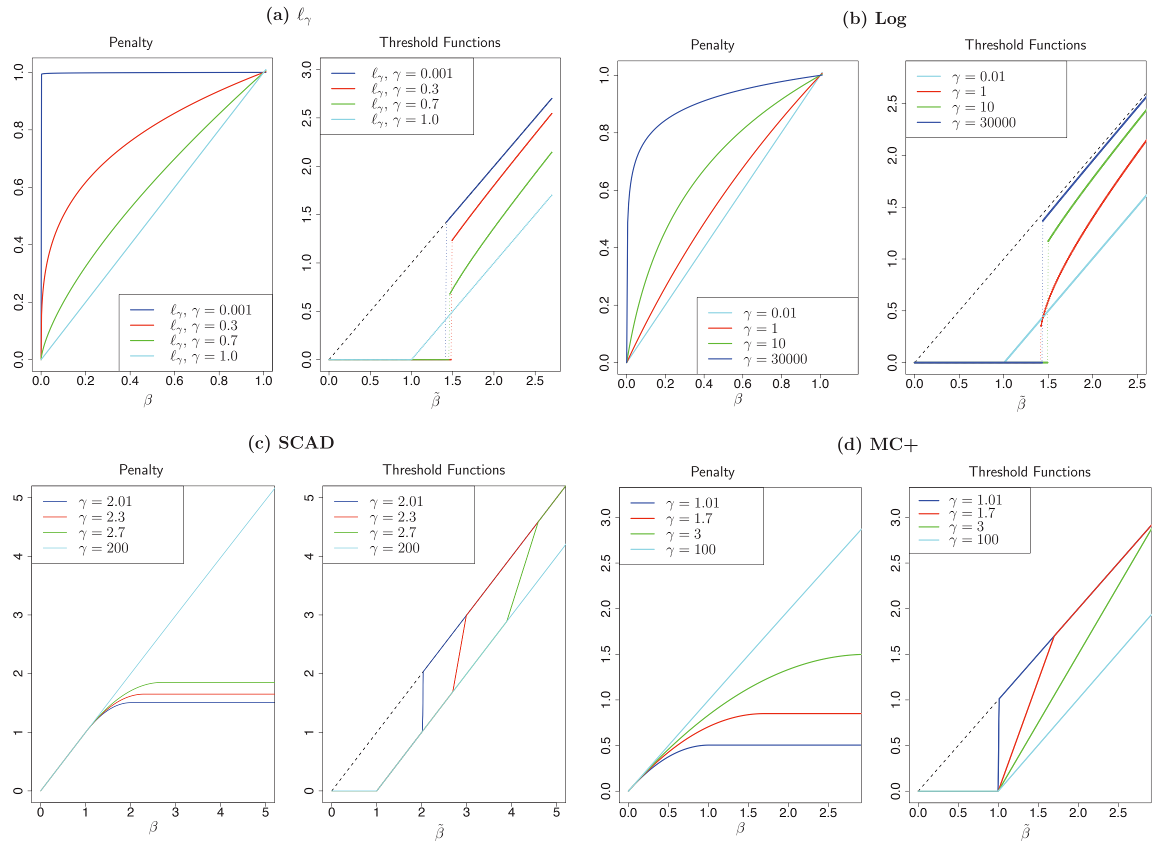

Figure 2.2: Non-Convex penalty families and their corresponding threshold functions. All are shown with $\lambda = 1$ and different values for γ.

to ℓ_0 ($\gamma \to \infty$).

(c) The SCAD penalty (Fan & Li 2001) is defined via

$$\begin{aligned} \frac{d}{dt}P(t;\lambda;\gamma) &= \mathbf{I}(t \le \lambda) + \frac{(\gamma\lambda - t)_+}{(\gamma - 1)\lambda}\mathbf{I}(t > \lambda) \text{ for } t > 0, \quad \gamma > 2 \\ P(t;\lambda;\gamma) &= P(-t;\lambda;\gamma) \\ P(0;\lambda;\gamma) &= 0 \end{aligned} \tag{2.11}$$

(d) The MC+ family of penalties (Zhang 2010) is defined by

$$\begin{aligned}\lambda P(t;\lambda;\gamma) &= \lambda\int_0^{|t|}(1-\frac{x}{\gamma\lambda})_+\,dx \\ &= \lambda(|t|-\frac{t^2}{2\lambda\gamma})\,\mathbf{I}(|t|<\lambda\gamma)+\frac{\lambda^2\gamma}{2}\,\mathbf{I}(|t|\geq\lambda\gamma).\end{aligned} \tag{2.12}$$

For each value of $\lambda > 0$ there is a continuum of penalties and threshold operators, varying from $\gamma \to \infty$ (soft threshold operator) to $\gamma \to 1+$ (hard threshold operator). The MC+ is a reparametrization of the *firm shrinkage* operator introduced by Gao & Bruce (1997) in the context of wavelet shrinkage.

Other examples of non-convex penalties include the transformed ℓ_1 penalty (Nikolova 2000) and the clipped ℓ_1 penalty (Zhang 2009).

Although each of these four families bridge ℓ_1 and ℓ_0, they have different properties. The two in the top row in Figure 2.2, for example, have discontinuous univariate threshold functions, which would cause instability in coordinate descent. The threshold operators for the ℓ_γ, log-penalty and the MC+ form a continuum between the soft and hard-thresholding functions. The family of SCAD threshold operators, although continuous, do not include $\mathrm{H}(\cdot,\lambda)$. We study some of these properties in more detail in Section 2.4.

2.3 *SparseNet*: coordinate-wise algorithm for constructing the entire regularization surface $\hat{\boldsymbol{\beta}}_{\lambda,\gamma}$

We now present our *SparseNet* algorithm for obtaining a family of solutions $\hat{\boldsymbol{\beta}}_{\gamma,\lambda}$ to (2.6). The $\mathbf{X}$ matrix is assumed to be standardized with each column having zero mean and unit ℓ_2 norm. For simplicity, we assume $\gamma = \infty$ corresponds to the LASSO and $\gamma = 1+$, the hard-thresholding members of the penalty families. The basic idea is as follows. For $\gamma = \infty$, we compute the exact solution path for $Q(\boldsymbol{\beta})$ as a function of λ using coordinate-descent.

These solutions are used as warm-starts for the minimization of $Q(\boldsymbol{\beta})$ at a smaller value of γ, corresponding to a more non-convex penalty. We continue in this fashion, decreasing γ, till we have the solutions paths across a grid of values for γ. The details are given in Algorithm 1.

Algorithm 1 *SparseNet*

1. Input a grid of increasing λ values $\Lambda = \{\lambda_1, \ldots, \lambda_L\}$, and a grid of increasing γ values $\Gamma = \{\gamma_1, \ldots, \gamma_K\}$, where γ_K indexes the LASSO penalty. Define λ_{L+1} such that $\hat{\boldsymbol{\beta}}_{\gamma_K, \lambda_{L+1}} = 0$.
2. For each value of $\ell \in \{L,\ L-1, \ldots, 1\}$ repeat the following
   (a) Initialize $\tilde{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_{\gamma_K, \lambda_{\ell+1}}$.
   (b) For each value of $k \in \{K, K-1, \ldots, 1\}$ repeat the following
      i. Cycle through the following one-at-a-time updates $j = 1, \ldots, p, 1, \ldots, p, \ldots$
      $$\tilde{\beta}_j = \mathrm{S}_{\gamma_k}\left(\sum_{i=1}^{n}(y_i - \tilde{y}_i^j)x_{ij}, \lambda_\ell\right), \tag{2.13}$$
      where $\tilde{y}_i^j = \sum_{k \neq j} x_{ik}\tilde{\beta}_k$, until the updates converge to $\boldsymbol{\beta}^*$.
      ii. Assign $\hat{\boldsymbol{\beta}}_{\gamma_k, \lambda_\ell} \leftarrow \boldsymbol{\beta}^*$.
   (c) Decrement k.
3. Decrement ℓ.
4. Return the two-dimensional solution surface $\hat{\boldsymbol{\beta}}_{\lambda,\gamma}$, $(\lambda, \gamma) \in \Lambda \times \Gamma$

In Section 2.4 we discuss certain properties of penalty functions and their threshold functions that are suited to this algorithm. We also discuss a particular form of *df* recalibration that provides attractive warm-starts and at the same time adds statistical meaning to the scheme.

We have found two variants of Algorithm 1 useful in practice:

- In computing the solution at (γ_k, λ_ℓ), the algorithm uses as a warm start the solution $\hat{\boldsymbol{\beta}}_{\gamma_{k+1}, \lambda_\ell}$. We run a parallel coordinate descent with warm start $\hat{\boldsymbol{\beta}}_{\gamma_k, \lambda_{\ell+1}}$, and then pick the solution with a smaller value for the objective function. This often leads to improved and smoother objective-value surfaces.

- It is sometimes convenient to have a different λ sequence for each value of γ—for example, our recalibrated penalties lead to such a scheme using a doubly indexed sequence $\lambda_{k\ell}$.

In Section 2.5 we implement *SparseNet* using the calibrated MC+ penalty, which enjoys all the properties outlined in Section 2.4.3 below. We show that the algorithm converges (Section 2.6) to a stationary point of $Q(\boldsymbol{\beta})$ for every (λ, γ).

2.4 Properties of families of non-convex penalties

Not all non-convex penalties are suitable for use with coordinate descent. Here we describe some desirable properties, and a recalibration suitable for our optimization Algorithm 1.

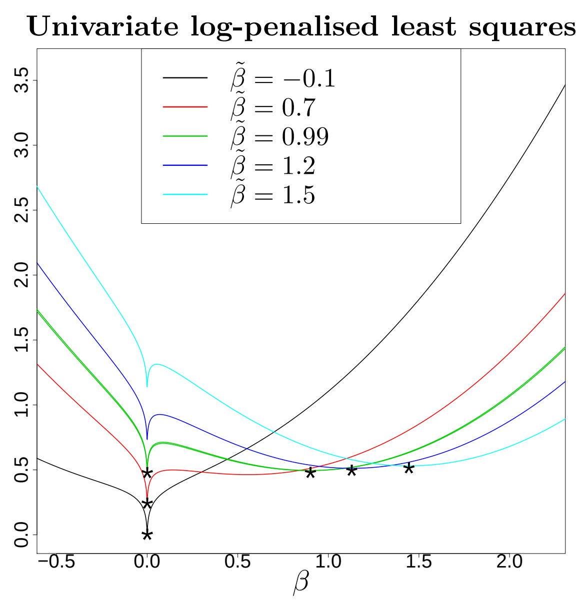

Figure 2.3: The penalized least-squares criterion (2.8) with the log-penalty (2.10) for $(\gamma, \lambda) = (500, 0.5)$ for different values of $\tilde{\beta}$. The "*" denotes the global minima of the functions. The "transition" of the minimizers, creates discontinuity in the induced threshold operators.

2.4.1 Effect of multiple minima in univariate criteria

Figure 2.3 shows the univariate penalized criterion (2.7) for the log-penalty (2.10) for certain choices of $\tilde{\beta}$ and a (λ, γ) combination. Here we see the non-convexity of $Q^1(\beta)$, and the transition (with $\tilde{\beta}$) of the global minimizers of the univariate functions. This causes the discontinuity of the induced threshold operators $\tilde{\beta} \mapsto \mathrm{S}_\gamma(\tilde{\beta}, \lambda)$, as shown in Figure 2.2(b). Multiple minima and consequent discontinuity appears in the $\ell_\gamma, \gamma < 1$ penalty as well, as seen in Figure 2.2(a).

It has been observed (Breiman 1996, Fan & Li 2001) that discontinuity of the threshold operators leads to increased variance (and hence poor risk properties) of the estimates. We observe that for the log-penalty (for example), coordinate-descent can produce multiple *limit points* (without converging) — creating statistical instability in the optimization procedure. We believe discontinuities such as this will naturally affect other optimization algorithms; for example those based on sequential convex relaxation such as MLLA. In the secondary Appendix 2.2.3 we see that MLLA gets stuck in a suboptimal local minimum, even for the univariate log-penalty problem. This phenomenon is aggravated for the ℓ_γ penalty.

Multiple minima and hence the discontinuity problem is not an issue in the case of the MC+ penalty or the SCAD (Figures 2.2(c,d)).

Our study of these phenomena leads us to conclude that if the *univariate* functions $Q^{(1)}(\beta)$ (2.7) are strictly convex, then the coordinate-wise procedure is well-behaved and converges to a stationary point. This turns out to be the case for the MC+ for $\gamma > 1$ and the SCAD penalties. Furthermore we see in Section 2.5 in (2.18) that this restriction on the MC+ penalty for γ still gives us the entire continuum of threshold operators from soft to hard thresholding.

Strict convexity of $Q^{(1)}(\beta)$ also occurs for the log-penalty for some choices of λ and γ, but not enough to cover the whole family. For example, the ℓ_γ family with $\gamma < 1$ does not qualify. This is not surprising since with $\gamma < 1$ it has an unbounded derivative at zero, and is well known to be unstable in optimization.

2.4.2 Effective *df* and nesting of *shrinkage-thresholds*

For a LASSO fit $\hat{\mu}_\lambda(x)$, λ controls the extent to which we (over)fit the data. This can be expressed in terms of the *effective degrees of freedom* of the estimator. For our non-convex penalty families, the *df* are influenced by γ as well. We propose to recalibrate our penalty families so that for fixed λ, the coordinate-wise *df* do not change with γ. This has important consequences for our pathwise optimization strategy.

For a linear model, *df* is simply the number of parameters fit. More generally, under an additive error model, *df* is defined by (Stein 1981, Efron et al. 2004)

$$df(\hat{\mu}_{\lambda,\gamma}) = \sum_{i=1}^{n} \mathrm{Cov}(\hat{\mu}_{\lambda,\gamma}(x_i), y_i)/\sigma^2, \tag{2.14}$$

where $\{(x_i, y_i)\}_1^n$ is the training sample and σ^2 is the noise variance. For a LASSO fit, the *df* is estimated by the number of nonzero coefficients (Zou et al. 2007). Suppose we compare the LASSO fit with k nonzero coefficients to an unrestricted least-squares fit in those k variables. The LASSO coefficients are shrunken towards zero, yet have the same *df*? The reason is the LASSO is "charged" for identifying the nonzero variables. Likewise a model with k coefficients chosen by best-subset regression has *df* greater than k (here we do not have exact formulas). Unlike the LASSO, these are not shrunk, but are being charged the extra *df* for the search.

Hence for both the LASSO and best subset regression we can think of the effective *df* as a function of λ. More generally, penalties corresponding to a smaller degree of non-convexity shrink more than those with a higher degree of non-convexity, and are hence charged less in *df* per nonzero coefficient. For the family of penalized regressions from ℓ_1 to ℓ_0, *df* is controlled by both λ and γ.

In this chapter we provide a re-calibration of the family of penalties $P(\cdot;\lambda;\gamma)$ such

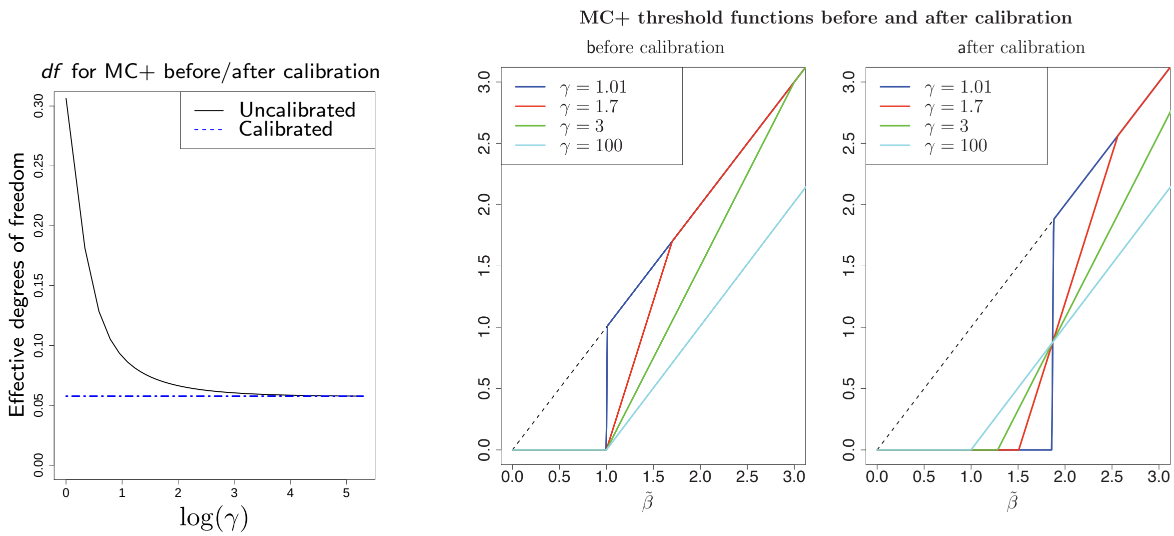

Figure 2.4: [Left] The *df* (solid line) for the (uncalibrated) MC+ threshold operators as a function of γ, for a fixed $\lambda = 1$. The dotted line shows the *df* after calibration. [Middle] Family of MC+ threshold functions, for different values of γ, before calibration. All have shrinkage threshold $\lambda_S = 1$. [Right] Calibrated versions of the same. The *shrinkage threshold* of the soft-thresholding operator is $\lambda = 1$, but as γ decreases, λ_S increases, forming a continuum between soft and hard thresholding.

that for every value of λ the coordinate-wise *df* across the entire continuum of γ values are approximately the same. Since we rely on coordinate-wise updates in the optimization procedures, we will ensure this by calibrating the *df* in the univariate thresholding operators. Details are given in Section 2.5.1 for a specific example.

We define the *shrinkage-threshold* λ_S of a thresholding operator as the largest (absolute) value that is set to zero. For the soft-thresholding operator of the LASSO, this is λ itself. However, the LASSO also shrinks the non-zero values toward zero. The hard-thresholding operator, on the other hand, leaves all values that survive its shrinkage threshold λ_H alone; in other words it fits the data more aggressively. So for the *df* of the soft and hard thresholding operators to be the same, the *shrinkage-threshold* λ_H for hard thresholding should be larger than the λ for soft thresholding. More generally, to maintain a constant *df* there should be a monotonicity (increase) in the *shrinkage-thresholds* λ_S as one moves across the family of threshold operators from the soft to the hard threshold operator. Figure 2.4

illustrates these phenomena for the MC+ penalty, before and after calibration.

Why the need for calibration? Because of the possibility of multiple stationary points in a non-convex optimization problem, good warm-starts are essential for avoiding sub-optimal solutions. The calibration assists in efficiently computing the doubly-regularized (γ, λ) paths for the coefficient profiles. For fixed λ we start with the exact LASSO solution (large γ). This provides an excellent warm start for the problem with a slightly decreased value for γ. This is continued in a gradual fashion as γ approaches the best-subset problem. The *df* calibration provides a natural path across neighboring solutions in the (λ, γ) space:

- λ_S increases slowly, decreasing the size of the active set;
- at the same time, the active coefficients are shrunk successively less.

We find that the calibration keeps the algorithm away from sub-optimal stationary points and accelerates the speed of convergence.

2.4.3 Desirable properties for a family of threshold operators for non-convex regularization

Consider the family of threshold operators

$$\mathrm{S}_\gamma(\cdot, \lambda) : \Re \to \Re \quad \gamma \in (\gamma_0, \gamma_1).$$

Based on our observations on the properties and irregularities of the different penalties and their associated threshold operators, we have compiled a list of properties we consider desirable:

1. $\gamma \in (\gamma_0, \gamma_1)$ should *bridge* the gap between soft and hard thresholding, with the following continuity at the end points

$$\mathrm{S}_{\gamma_1}(\tilde{\beta}, \lambda) = \mathrm{sgn}(\tilde{\beta})(\tilde{\beta} - \lambda)_+ \text{ and } \mathrm{S}_{\gamma_0}(\tilde{\beta}; \lambda) = \tilde{\beta}\mathbf{I}(|\tilde{\beta}| \geq \lambda_H) \tag{2.15}$$

where λ and λ_H correspond to the *shrinkage thresholds* of the soft and hard threshold operators respectively.

2. λ should control the effective *df* in the family $\mathrm{S}_\gamma(\cdot, \lambda)$; for fixed λ, the effective *df* of $\mathrm{S}_\gamma(\cdot, \lambda)$ for all values of γ should be the same.

3. For every fixed λ, there should be a strict nesting (increase) of the *shrinkage thresholds* λ_S as γ decreases from γ_1 to γ_0.

4. The map $\tilde{\beta} \mapsto \mathrm{S}_\gamma(\tilde{\beta}, \lambda)$ should be continuous.

5. The univariate penalized least squares function $Q^{(1)}(\beta)$ (2.7) should be convex for every $\tilde{\beta}$. This ensures that coordinate-wise procedures converge to a stationary point. In addition this implies continuity of $\tilde{\beta} \mapsto \mathrm{S}_\gamma(\tilde{\beta}, \lambda)$ in the previous item.

6. The function $\gamma \mapsto \mathrm{S}_\gamma(\cdot, \lambda)$ should be continuous on $\gamma \in (\gamma_0, \gamma_1)$. This assures a smooth transition as one moves across the family of penalized regressions, in constructing the family of regularization paths.

We believe these properties are necessary for a meaningful analysis for any generic non-convex penalty. Enforcing them will require re-parametrization and some restrictions in the family of penalties (and threshold operators) considered. In terms of the four families discussed in Section 2.2:

- The threshold operator induced by the SCAD penalty does not encompass the entire continuum from the soft to hard thresholding operators (all the others do).

- None of the penalties satisfy the nesting property of the shrinkage thresholds (item 3) or the degree of freedom calibration (item 2). In Section 2.5.1 we recalibrate the MC+ penalty so as to achieve these.

- The threshold operators induced by ℓ_γ and Log-penalties are not continuous (item 4); they are for MC+ and SCAD.

- $Q^{(1)}(\beta)$ is strictly convex for both the SCAD and the MC+ penalty. $Q^{(1)}(\beta)$ is non-convex for *every* choice of $(\lambda > 0, \gamma)$ for the ℓ_γ penalty and *some* choices of $(\lambda > 0, \gamma)$ for the log-penalty.

In this chapter we explore the details of our approach through the study of the MC+ penalty; indeed, it is the only one of the four we consider for which all the properties above are achievable.

2.5 Illustration via the MC+ penalty

Here we give details on the recalibration of the MC+ penalty, and the algorithm that results. The MC+ penalized univariate least-squares objective criterion is

$$Q^1(\beta) = \tfrac{1}{2}(\beta - \tilde{\beta})^2 + \lambda \int_0^{|\beta|} \left(1 - \frac{x}{\gamma\lambda}\right)_+ dx. \tag{2.16}$$

This can be shown to be convex for $\gamma \geq 1$, and non-convex for $\gamma < 1$. The minimizer for $\gamma > 1$ is a piecewise-linear thresholding function (see Figure 2.4, middle) given by

$$\mathrm{S}_\gamma(\tilde{\beta}, \lambda) = \begin{cases} 0 & \text{if } |\tilde{\beta}| \leq \lambda; \\ \operatorname{sgn}(\tilde{\beta}) \left(\frac{(|\tilde{\beta}| - \lambda}{1 - \frac{1}{\gamma}} \right) & \text{if } \lambda < |\tilde{\beta}| \leq \lambda\gamma; \\ \tilde{\beta} & \text{if } |\tilde{\beta}| > \lambda\gamma. \end{cases} \tag{2.17}$$

Observe that in (2.17), for fixed $\lambda > 0$,

$$\begin{aligned} \text{as } \gamma \to 1+, \qquad & \mathrm{S}_\gamma(\tilde{\beta}, \lambda) \to \mathrm{H}(\tilde{\beta}, \lambda), \\ \text{as } \gamma \to \infty, \qquad & \mathrm{S}_\gamma(\tilde{\beta}, \lambda) \to \mathrm{S}(\tilde{\beta}, \lambda). \end{aligned} \tag{2.18}$$

Hence $\gamma_0 = 1+$ and $\gamma_1 = \infty$. It is interesting to note here that the hard-threshold operator, which is conventionally understood to arise from a highly non-convex ℓ_0 penalized criterion

(2.9), can be equivalently obtained as the limit of a sequence $\{\mathrm{S}_\gamma(\tilde{\beta}, \lambda)\}_{\gamma>1}$ where each threshold operator is the solution of a *convex criterion* (2.16) for $\gamma > 1$. For a fixed λ, this gives a family $\{\mathrm{S}_\gamma(\tilde{\beta}, \lambda)\}_\gamma$ with the soft and hard threshold operators as its two extremes.

2.5.1 Calibrating MC+ for *df*

For a fixed λ, we would like all the thresholding functions $\mathrm{S}_\gamma(\tilde{\beta}, \lambda)$ to have the same *df*; this will require a reparametrization. It turns out that this is tractable for the MC+ threshold function.

Consider the following univariate regression model

$$y_i = x_i\beta + \epsilon_i,\ i = 1, \ldots, n,\ \epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2). \tag{2.19}$$

Without loss of generality, assume the x_i's have sample mean zero and sample variance one[3]. The *df* for the threshold operator $\mathrm{S}_\gamma(\cdot, \lambda)$ is defined by (2.14) with

$$\hat{\boldsymbol{\mu}}_{\gamma,\lambda} = \mathbf{x} \cdot \mathrm{S}_\gamma(\tilde{\beta}, \lambda) \tag{2.20}$$

and $\tilde{\beta} = \sum_i x_i y_i / \sum x_i^2$. The following theorem gives an explicit expression of the *df*.

Theorem 1. *For the model described in (2.19), the* df *of* $\hat{\boldsymbol{\mu}}_{\gamma,\lambda}$ *(2.20) is given by:*

$$df(\hat{\boldsymbol{\mu}}_{\gamma,\lambda}) = \frac{\lambda\gamma}{\lambda\gamma - \lambda}\mathrm{Pr}(\lambda \leq |\tilde{\beta}| < \lambda\gamma) + \mathrm{Pr}(|\tilde{\beta}| > \lambda\gamma) \tag{2.21}$$

where these probabilities are to be calculated under the law $\tilde{\beta} \sim N(\beta, \sigma^2/n)$

Proof. We make use of Stein's unbiased risk estimation (Stein 1981, SURE) result. It states that if $\hat{\boldsymbol{\mu}} : \Re^n \to \Re^n$ is an almost differentiable function of $\mathbf{y}$, and $\mathbf{y} \sim N(\boldsymbol{\mu}, \sigma^2\mathbf{I}_n)$, then

[3] The x_i's are assumed to be non-random

there is a simplification to the $df(\hat{\boldsymbol{\mu}})$ formula (2.14)

$$\begin{aligned} df(\hat{\boldsymbol{\mu}}) &= \sum_{i=1}^{n} \mathrm{Cov}(\hat{\mu}_i, y_i)/\sigma^2 \\ &= E(\nabla \cdot \hat{\boldsymbol{\mu}}), \end{aligned} \tag{2.22}$$

where $\nabla \cdot \hat{\boldsymbol{\mu}} = \sum_{i=1}^{n} \partial\hat{\mu}_i/\partial y_i$. Proceeding along the lines of proof in Zou et al. (2007), the function $\hat{\boldsymbol{\mu}}_{\gamma,\lambda} : \Re \to \Re$ defined in (2.20) is uniformly Lipschitz (for every $\lambda \geq 0, \gamma > 1$), and hence almost everywhere differentiable. The result follows easily from the definition (2.17) and that of $\tilde{\beta}$, which leads to (2.21). □

As $\gamma \to \infty$, $\hat{\boldsymbol{\mu}}_{\gamma,\lambda} \to \hat{\boldsymbol{\mu}}_{\lambda}$, and from (2.21), we see that

$$\begin{aligned} df(\hat{\boldsymbol{\mu}}_{\gamma,\lambda}) &\longrightarrow \Pr(|\tilde{\beta}| > \lambda) \qquad (2.23) \\ &= E\left(I(|\tilde{\beta}| > 0)\right) \end{aligned}$$

which corresponds to the expression obtained by Zou et al. (2007) for the *df* of the LASSO in the univariate case.

Corollary 1. *The* df *for the hard-thresholding function* $\mathrm{H}(\tilde{\beta}, \lambda)$ *is given by*

$$df(\hat{\boldsymbol{\mu}}_{1+,\lambda}) = \lambda\phi^*(\lambda) + \Pr(|\tilde{\beta}| > \lambda) \tag{2.24}$$

where ϕ^* is taken to be the p.d.f. of the absolute value of a normal random variable with mean β and variance σ^2/n.

For the hard threshold operator, Stein's simplified formula does not work, since the corresponding function $\boldsymbol{y} \mapsto \mathrm{H}(\tilde{\beta}, \lambda)$ is not almost differentiable. But observing from (2.21) that

$$df(\hat{\boldsymbol{\mu}}_{\gamma,\lambda}) \to \lambda\phi^*(\lambda) + \Pr(|\tilde{\beta}| > \lambda), \text{ and } \mathrm{S}_{\gamma}(\tilde{\beta}, \lambda) \to \mathrm{H}(\tilde{\beta}, \lambda) \text{ as } \gamma \to 1+ \tag{2.25}$$

we get an expression for *df* as stated.

These expressions are consistent with simulation studies based on Monte Carlo estimates of *df*. Figure 2.4[Left] shows *df* as a function of γ for a fixed value $\lambda = 1$ for the uncalibrated MC+ threshold operators $\mathrm{S}_\gamma(\cdot, \lambda)$. For the figure we used $\beta = 0$ and $\sigma^2 = 1$.

Re-parametrization of the MC+ penalty

We argued in Section 2.4.2 that for the *df* to remain constant for a fixed λ and varying γ, the shrinkage threshold $\lambda_S = \lambda_S(\lambda, \gamma)$ should increase as γ moves from γ_1 to γ_0. Theorem 3 formalizes this observation.

Hence for the purpose of *df* calibration, we re-parametrize the family of penalties as follows:

$$\lambda P^*(|t|; \lambda; \gamma) = \lambda_S(\lambda, \gamma) \int_0^{|t|} \left(1 - \frac{x}{\gamma \lambda_S(\lambda, \gamma)}\right)_+ dx. \tag{2.26}$$

With the re-parametrized penalty, the thresholding function is the obvious modification of (2.17):

$$\begin{aligned} \mathrm{S}^*_\gamma(\tilde\beta, \lambda) &= \arg\min_\beta \{\tfrac{1}{2}(\beta - \tilde\beta)^2 + \lambda P^*(|\beta|; \lambda; \gamma)\} \\ &= \mathrm{S}_\gamma(\tilde\beta, \lambda_S(\lambda, \gamma)) \end{aligned} \tag{2.27}$$

Similarly for the *df* we simply plug into the formula (2.21)

$$df(\hat{\boldsymbol{\mu}}^*_{\gamma,\lambda}) = \frac{\gamma \lambda_S(\lambda, \gamma)}{\gamma \lambda_S(\lambda, \gamma) - \lambda_S(\lambda, \gamma)} \Pr(\lambda_S(\lambda, \gamma) \le |\tilde\beta| < \gamma \lambda_S(\lambda, \gamma)) + \Pr(|\tilde\beta| > \gamma \lambda_S(\lambda, \gamma)) \tag{2.28}$$

where $\hat{\boldsymbol{\mu}}^*_{\gamma,\lambda} = \hat{\boldsymbol{\mu}}_{\gamma, \lambda_S(\lambda,\gamma)}$.

To achieve a constant *df*, we require the following to hold for $\lambda > 0$:

- The shrinkage-threshold for the soft threshold operator is $\lambda_S(\lambda, \gamma = \infty) = \lambda$ and hence $\hat{\boldsymbol{\mu}}^*_\lambda = \hat{\boldsymbol{\mu}}_\lambda$ (a boundary condition in the calibration).

- $df(\hat{\boldsymbol{\mu}}^*_{\gamma,\lambda}) = df(\hat{\boldsymbol{\mu}}_\lambda) \; \forall \; \gamma > 1$.

The definitions for *df* depend on β and σ^2/n. Since the notion of *df* centers around variance, we use the null model with $\beta = 0$. We also assume without loss of generality that $\sigma^2/n = 1$, since this gets absorbed into λ.

Theorem 2. *For the calibrated MC+ penalty to achieve constant* df*, the shrinkage-threshold $\lambda_S = \lambda_S(\lambda, \gamma)$ must satisfy the following functional relationship*

$$\Phi(\gamma\lambda_S) - \gamma\Phi(\lambda_S) = -(\gamma - 1)\Phi(\lambda), \tag{2.29}$$

where Φ is the standard normal cdf and ϕ the pdf of the same.

Theorem 2 is proved in the secondary Appendix 2.2.1, using (2.28) and the boundary constraint. The next theorem establishes some properties of of the map $(\lambda, \gamma) \mapsto (\lambda_S(\lambda, \gamma), \gamma\lambda_S(\lambda, \gamma))$, including the important nesting of shrinkage thresholds.

Theorem 3. *For a fixed λ,*

(a) $\gamma\lambda_S(\lambda, \gamma)$ *is increasing as a function of* γ.

(b) $\lambda_S(\lambda, \gamma)$ *is decreasing as a function of* γ

Note that as γ increases, we approach the LASSO; see Figure 2.4 (right). Both these relationships can be derived from the functional equation (2.29). Theorem 3 is proved in the secondary Appendix 2.2.1.

Efficient computation of *shrinkage thresholds*

In order to implement the calibration for MC+, we need an efficient method for evaluating $\lambda_S(\lambda, \gamma)$—i.e. solving equation (2.29). For this purpose we propose a simple parametric form for λ_S based on some of its required properties: the monotonicity properties just described, and the *df* calibration. We simply give the expressions here, and leave the details for the secondary Appendix 2.2.2:

$$\lambda_S(\lambda,\gamma) = \lambda_H(\frac{1-\alpha^*}{\gamma} + \alpha^*) \tag{2.30}$$

where

$$\alpha^* = \lambda_H^{-1}\Phi^{-1}(\Phi(\lambda_H) - \lambda_H\phi(\lambda_H)), \quad \lambda = \lambda_H\alpha^* \tag{2.31}$$

The above approximation turns out to be a reasonably good estimator for all practical purposes, achieving a calibration of *df* within an accuracy of five percent, uniformly over all (γ, λ). The approximation can be improved further, if we take this estimator as the starting point, and obtain recursive updates for $\lambda_S(\lambda,\gamma)$. Details of this approximation along with the algorithm are explained in the secondary Appendix 2.2.2. Numerical studies show that a few iterations of this recursive computation can improve the degree of accuracy of calibration up to an error of 0.3 percent; Figure 2.4 [Left] was produced using this approximation. Figure 2.5 shows a typical pattern of recalibration for the example we present in Figure 2.7 in Section 2.8.

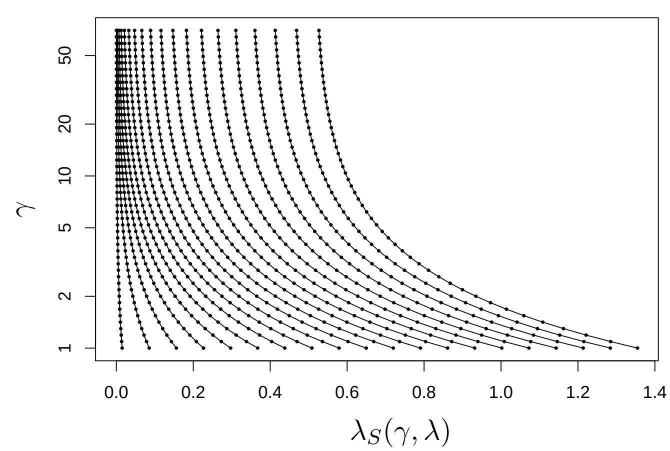

Figure 2.5: Recalibrated values of λ via $\lambda_S(\gamma,\lambda)$ for the MC+ penalty. The values of λ at the top of the plot correspond to the LASSO. As γ decreases, the calibration increases the shrinkage threshold to achieve a constant univariate df.

As noted below Algorithm 1 this leads to a lattice of values $\lambda_{k,\ell}$.

2.6 Convergence Analysis

In this section we present results on the convergence of Algorithm 1. Tseng & Yun (2009) show convergence of coordinate-descent for functions that can be written as the sum of a smooth function (loss) and a separable non-smooth convex function (penalty). This is not directly applicable to our case as the penalty is non-convex.

Denote the coordinate-wise updates $\boldsymbol{\beta}^{k+1} = \mathrm{S}_{\mathrm{cw}}(\boldsymbol{\beta}^k),\ k = 1, 2, \ldots$ with

$$\beta_j^{k+1} = \arg\min_u \ Q(\beta_1^{k+1}, \ldots, \beta_{j-1}^{k+1}, u, \beta_{j+1}^k, \ldots, \beta_p^k), j = 1, \ldots, p. \tag{2.32}$$

Theorem 4 establishes that under certain conditions, *SparseNet* always converges to a minimum of the objective function; conditions that are met, for example, by the SCAD and MC+ penalties (for suitable γ).

Theorem 4. *Consider the criterion in (2.6), where the given data $(y, \boldsymbol{X})$ lies on a compact set and no column of $\boldsymbol{X}$ is degenerate (ie multiple of the unit vector). Suppose the penalty $\lambda P(t; \lambda; \gamma) \equiv P(t)$ satisfies $P(t) = P(-t)$, $P'(|t|)$ is non-negative, uniformly bounded and $\inf_t P''(|t|) > -1$; where $P'(|t|)$ and $P''(|t|)$ are the first and second derivatives (assumed to exist) of $P(|t|)$ wrt $|t|$.*

Then the univariate maps $\beta \mapsto Q^{(1)}(\beta)$ are strictly convex and the sequence of coordinate-updates $\{\boldsymbol{\beta}^k\}_k$ converge to a minimum of the function $Q(\boldsymbol{\beta})$.

Note that the condition on the data $(y, \mathbf{X})$ is a mild assumption (as it is necessary for the variables to be standardized). Since the columns of $\mathbf{X}$ are mean-centered, the non-degeneracy assumption is equivalent to assuming that no column is identically zero.

The proof is provided in Appendix 2.A.1. Lemma 1 and Lemma 3 under milder regularity conditions, establish that every limit point of the sequence $\{\boldsymbol{\beta}^k\}$ is a stationary point of

$Q(\boldsymbol{\beta})$.

Remark 1. *Note that Theorem 4 includes the case where the penalty function $P(t)$ is convex in t.*

2.7 Simulation Studies

In this section we compare the simulation performance of a number of different methods with regard to a) prediction error, b) the number of non-zero coefficients in the model, and c) "misclassification error" of the variables retained. The methods we compare are *SparseNet*, Local Linear Approximation (LLA) and Multi-stage Local Linear Approximation (MLLA) (Zou & Li 2008, Zhang 2009, Candes et al. 2008), LASSO, forward-stepwise regression and best-subset selection. Note that *SparseNet*, LLA and MLLA are all optimizing the same MC+ penalized criterion.

We assume a linear model $Y = X\beta + \varepsilon$ with multivariate Gaussian predictors X and Gaussian errors. The Signal-to-Noise Ratio (SNR) and Standardized Prediction Error (SPE) are defined as

$$\text{SNR} = \frac{\sqrt{\boldsymbol{\beta}^T \Sigma \boldsymbol{\beta}}}{\sigma}, \quad \text{SPE} = \frac{E(\mathbf{y} - \mathbf{x}\hat{\beta})^2}{\sigma^2} \tag{2.33}$$

The minimal achievable value for SPE is 1 (the Bayes error rate). For each model, the *optimal* tuning parameters — λ for the penalized regressions, subset size for best-subset selection and stepwise regression—are chosen based on minimization of the prediction error on a separate large validation set of size 10K. We use SNR=3 in all the examples.

Since a primary motivation for considering non-convex penalized regressions is to mimic the behavior of best subset selection, we compare its performance with best-subset for small $p = 30$. For notational convenience $\Sigma(\rho; m)$ denotes a $m \times m$ matrix with 1's on the diagonal, and ρ's on the off-diagonal.

We consider the following examples:

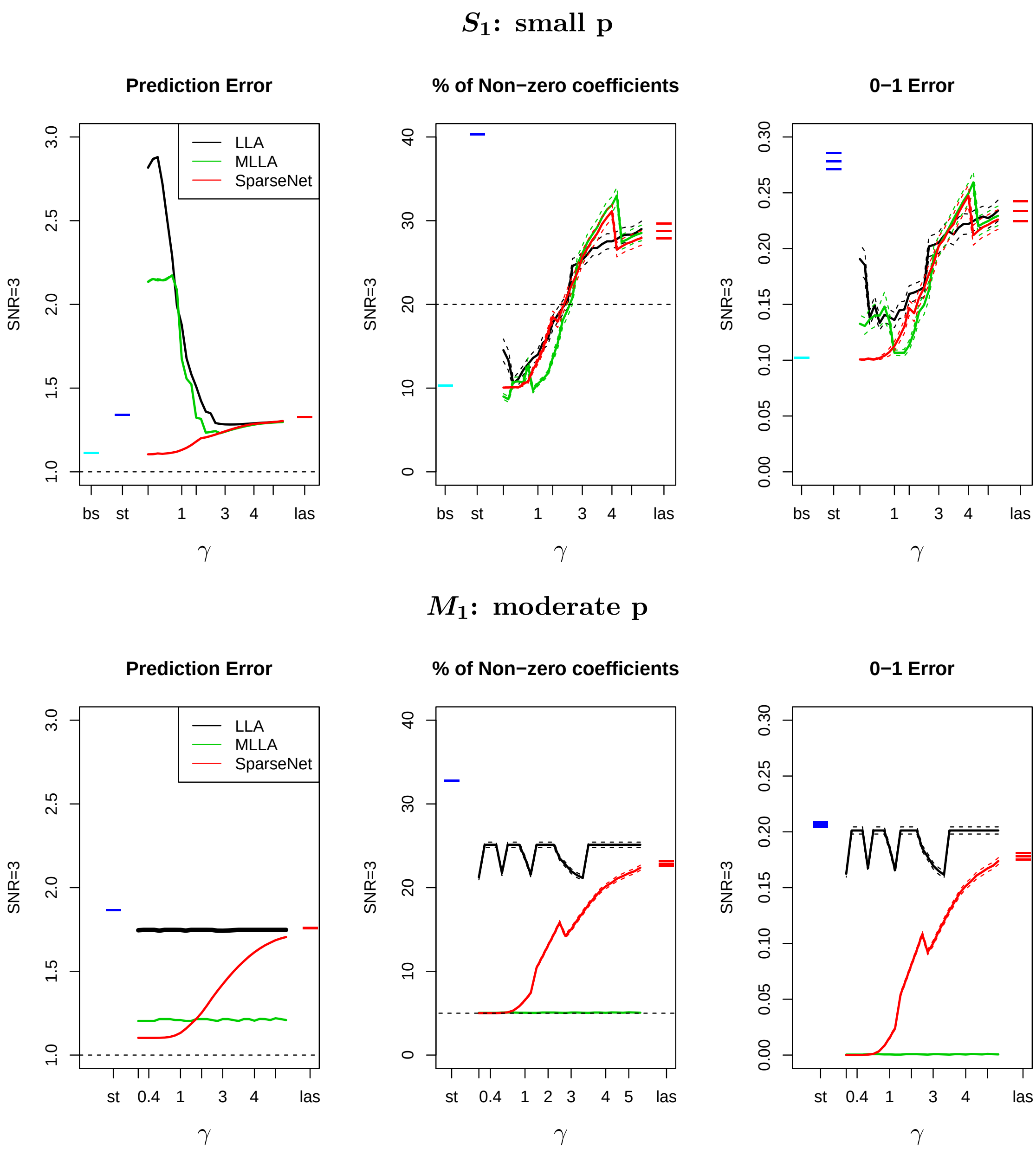

Figure 2.6: Examples $\boldsymbol{S_1}$ and $\boldsymbol{M_1}$. The three columns of plots represent prediction error, recovered sparsity and zero-one loss (all with standard errors over the 25 runs). Plots show the recalibrated MC+ family for different values of γ (on the log-scale) via *SparseNet*, LLA, MLLA, LASSO (las), step-wise (st) and best-subset (bs) (for $\boldsymbol{S_1}$ only)

$\boldsymbol{S_1}$: $n = 35,\ p = 30,\ \Sigma^{S_1} = \Sigma(0.4; p)$ and $\boldsymbol{\beta}^{S_1} = (0.03, 0.07, 0.1, 0.9, 0.93, 0.97, \mathbf{0}_{1\times 24})$.

$\boldsymbol{M_1}$: $n = 100,\ p = 200,\ \Sigma^{M_1} = \{0.7^{|i-j|}\}_{1\le i,j\le p}$ and $\boldsymbol{\beta}^{M_1}$ has 10 non-zeros such that $\beta^{M_1}_{20i+1} = 1,\ i = 0, 1, \ldots, 9$; and $\beta^{M_1}_i = 0$ otherwise.

In example $\boldsymbol{M_1}$ best-subset was not feasible and hence was omitted. The starting point for both LLA and MLLA were taken as a vector of all ones (Zhang 2009). Results are shown in Figure 2.6 with discussion in Section 2.7.1. These two examples, chosen from a battery of similar examples, show that *SparseNet* compares very favorably with its competitors.

We now consider some larger examples and study the performance of *SparseNet* varying γ.

$\boldsymbol{M_1}(5)$: $n = 500,\ p = 1000$, $\Sigma^{M_1(5)} = \text{blockkdiag}(\Sigma^{M_1}, \ldots, \Sigma^{M_1})$ and $\boldsymbol{\beta}^{M_1(5)} = (\boldsymbol{\beta}^{M_1}, \ldots, \boldsymbol{\beta}^{M_1})$ (five blocks).

$\boldsymbol{M_1}(10)$: $n = 500,\ p = 2000$ (same as above with ten blocks instead of five).

$\boldsymbol{M_2}(5)$: $n = 500,\ p = 1000,\ \Sigma^{M_2(5)} = \text{blockdiag}(\Sigma(0.5, 200), \ldots, \Sigma(0.5, 200))$ and $\boldsymbol{\beta}^{M_2(5)} = (\boldsymbol{\beta}^{M_2}, \ldots, \boldsymbol{\beta}^{M_2})$ (five blocks). Here $\boldsymbol{\beta}^{M_2} = (\beta_1, \beta_2, \ldots, \beta_{10}, \mathbf{0}_{1\times 190})$ is such that the first ten coefficients form an equi-spaced grid on $[0, 0.5]$.

$\boldsymbol{M_2}(10)$: $n = 500,\ p = 2000$, and is like $\boldsymbol{M_2}(5)$ with ten blocks.

The results are summarized in Table 2.1, with discussion in Section 2.7.1.

2.7.1 Discussion of Simulation Results

In both $\boldsymbol{S_1}$ and $\boldsymbol{M_1}$, the aggressive non-convex *SparseNet* out-performs its less aggressive counterparts in terms of prediction error and variable selection. Due to the correlation among the features, the LASSO and the less aggressive penalties estimate a considerable proportion of zero coefficients as non-zero. *SparseNet* estimates for $\gamma \approx 1$ are almost identical to the best-subset procedure in $\boldsymbol{S_1}$. Step-wise performs worse in both cases. LLA and

	log(γ) values			
Example: $\boldsymbol{M_1}(5)$	3.92	1.73	1.19	0.10
SPE	$1.6344_{(9.126e-4)}$	$1.3194_{(5.436e-4)}$	$1.2015_{(4.374e-4)}$	$1.1313_{(3.654e-4)}$
% of non-zeros [5]	$16.370_{(0.1536)}$	$13.255_{(0.1056)}$	$8.195_{(0.0216)}$	$5.000_{(0.0)}$
$0-1$ error	$.1137_{(1.536e-3)}$	$.0825_{(1.056e-3)}$	$.0319_{(.724e-3)}$	$.0002_{(.087e-3)}$
Example: $\boldsymbol{M_2}(5)$				
SPE	$1.3797_{(0.486e-3)}$	$1.467_{(0.531e-3)}$	$1.499_{(7.774e-3)}$	$1.7118_{(0.918e-3)}$
% of non-zeros [5]	$21.435_{(0.1798)}$	$9.900_{(0.1156)}$	$6.74_{(0.097)}$	$2.490_{(0.0224)}$
$0-1$ error	$0.18445_{(1.63e-3)}$	$0.08810_{(1.428e-3)}$	$0.0592_{(0.841e-3)}$	$0.03530_{(0.571e-3)}$
Example: $\boldsymbol{M_1}(10)$				
SPE	$3.6819_{(0.58e-2)}$	$4.4838_{(0.99e-2)}$	$4.653_{(1.033e-2)}$	$5.4729_{(2.006e-2)}$
% of non-zeros [5]	$22.987_{(0.103)}$	$11.9950_{(0.075)}$	$8.455_{(0.0485)}$	$6.2625_{(0.0378)}$
$0-1$ error	$0.1873_{(1.172e-3)}$	$0.0870_{(1.113e-3)}$	$0.0577_{(1.191e-3)}$	$0.0446_{(1.473e-3)}$
Example: $\boldsymbol{M_2}(10)$				
SPE	$1.693_{(.906e-3)}$	$1.893_{(.756e-3)}$	$2.318_{(1.383e-3)}$	$2.631_{(1.638e-2)}$
% of non-zeros [5]	$15.4575_{(0.128)}$	$8.2875_{(0.066)}$	$5.6625_{(0.059)}$	$2.22_{(0.0214)}$
$0-1$ error	$0.1375_{(1.384e-3)}$	$0.0870_{(1.1e-3)}$	$0.0678_{(0.554e-3)}$	$0.047_{(0.486e-3)}$

Table 2.1: Table showing standardized prediction error (SPE), percentage of non-zeros and zero-one error in the recovered model via *SparseNet*, for different problem instances. Values within parenthesis (in subscripts) denote the standard errors, and the true number of non-zero coefficients in the model are in square braces. Results are averaged over 25 runs.

MLLA show similar behavior in $\boldsymbol{S_1}$, but are inferior to *SparseNet* in predictive performance for smaller values of γ. This shows that MLLA and *SparseNet* reach very different local optima. In $\boldsymbol{M_1}$ LLA/MLLA seem to show similar behavior across varying degrees of non-convexity. This is undesirable behavior, and is probably because MLLA/LLA gets stuck in a local minima. MLLA does perfect variable selection and shows good predictive accuracy.

In $\boldsymbol{M_1}(5)$, $(n = 500, p = 1000)$ the prediction error decreases steadily with decreasing γ, the variable selection properties improve as well. In $\boldsymbol{M_1}(10)$ $(n = 500, p = 2000)$ the prediction error increases overall, and there is a trend reversal — with the more aggressive non-convex penalties performing worse. The variable selection properties of the aggressive penalties, however, are superior to its counterparts. The less aggressive non-convex selectors include a larger number of variables with high shrinkage, and hence perform well in predictive accuracy.

In $\boldsymbol{M_2}(5)$ and $\boldsymbol{M_2}(10)$, the prediction accuracy decreases marginally with increasing γ. However, as before, the variable selection properties of the more aggressive non-convex penalties are far better.

In summary, *SparseNet* is able to mimic the prediction performance of best subset regression in these examples. In the high-p settings, it repeats this behavior, mimicking the best, and is the out-right winner in several situations since best-subset regression is not available. In situations where the less aggressive penalties do well, *SparseNet* often reduces the number of variables for comparable prediction performance, by picking a γ in the interior of its range. The solutions of LLA and MLLA are quite different from the *SparseNet* and they often show similar performances across γ (as also pointed out in Candes et al. (2008) and Zou & Li (2008)).

It appears that the main differences among the different strategies MLLA, LLA and *SparseNet* lie in their roles of optimizing the objective $Q(\boldsymbol{\beta})$. We study this from a well grounded theoretical framework in Section 2.8, simulation experiments in Section 2.8.1 and further examples in Section 2.2.3 in the secondary appendix.

2.8 Other methods of optimization

We shall briefly review some of the state-of-the art methods proposed for optimization with general non-convex penalties.

Fan & Li (2001) used a local quadratic approximation (LQA) of the SCAD penalty. The method can be viewed as a majorize-minimize algorithm which repeatedly performs a weighted ridge regression. LQA gets rid of the non-singularity of the penalty at zero, depends heavily on the initial choice $\boldsymbol{\beta}^{\mathbf{0}}$, and is hence suboptimal (Zou & Li 2008) in searching for sparsity.

The MC+ algorithm of Zhang (2010) cleverly tracks multiple local minima of the objective function and seeks a solution to attain desirable statistical properties. The algorithm is complex, and since it uses a LARS-type update, is potentially slower than coordinate-wise procedures.

Friedman (2008) proposes a path-seeking algorithm for general non-convex penalties. Except in simple cases, it is unclear what criterion of the form "loss+penalty" it optimizes.

Multi-stage Local Linear Approximation. Zou & Li (2008), Zhang (2009) and Candes et al. (2008) propose majorize-minimize (MM) algorithms for minimization of $Q(\boldsymbol{\beta})$ (2.6), and consider the re-weighted ℓ_1 criterion

$$Q(\boldsymbol{\beta};\boldsymbol{\beta}^k) = \text{Constant} + \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} P'(|\hat{\beta}_j^k|;\lambda;\gamma)|\beta_j| \tag{2.34}$$

for some starting point $\boldsymbol{\beta}^1$. If $\boldsymbol{\beta}^{k+1}$ is obtained by minimizing (2.34)— we denote the update via the map $\boldsymbol{\beta}^{k+1} := M_{\text{lla}}(\boldsymbol{\beta}^k)$. There is currently a lot of attention payed to MLLA in the literature, so we analyze and compare it to our coordinate-wise procedure in more detail.

We have observed (Sections 2.8.1 and 2.2.3) that the starting point is critical; see also Candes et al. (2008) and Zhang (2010). Zou & Li (2008) suggest stopping at $k = 2$ after starting with an "optimal" estimator, for example, the least-squares estimator when $p < n$.

Choosing an optimal initial estimator $\boldsymbol{\beta}^1$ is essentially equivalent to knowing a-priori the zero and non-zero coefficients of $\boldsymbol{\beta}$. The adaptive LASSO of Zou (2006) is very similar in spirit to this formulation, though the weights (depending upon a good initial estimator) are allowed to be more general than derivatives of the penalty (2.34).

Convergence properties of *SparseNet* and MLLA can be analyzed via properties of fixed points of the maps $\mathrm{S}_{\text{cw}}(\cdot)$ and $M_{\text{lla}}(\cdot)$ respectively. If the sequences produced by $M_{\text{lla}}(\cdot)$ or $\mathrm{S}_{\text{cw}}(\cdot)$ converge, they correspond to stationary points of the function Q. Fixed points of the maps $M_{\text{lla}}(\cdot)$ and $\mathrm{S}_{\text{cw}}(\cdot)$ correspond to convergence of the sequence of updates.

The convergence analysis of MLLA is based on the fact that $Q(\boldsymbol{\beta};\boldsymbol{\beta}^k)$ majorizes $Q(\boldsymbol{\beta})$. If $\boldsymbol{\beta}^k$ is the sequence produced via MLLA, then $\{Q(\boldsymbol{\beta}^k)\}_{k\geq 1}$ converges. This does not address the convergence properties of $\{\boldsymbol{\beta}^k\}_{k\geq 1}$ nor properties of its limit points. Zou & Li (2008) point out that if the map M_{lla} satisfies $Q(\boldsymbol{\beta}) = Q(M_{\text{lla}}(\boldsymbol{\beta}))$ for limit points of the sequence $\{\boldsymbol{\beta}^k\}_{k\geq 1}$, then the limit points are stationary points of the function $Q(\boldsymbol{\beta})$. It appears, however, that solutions of $Q(\boldsymbol{\beta}) = Q(M_{\text{lla}}(\boldsymbol{\beta}))$ are not easy to characterize by explicit regularity conditions on the penalty P. The analysis does not address the fate of all the limit points of the sequence $\{\boldsymbol{\beta}^k\}_{k\geq 1}$ in case the sufficient conditions of Proposition 1 in Zou & Li (2008) fail to hold true. Our analysis in Section 2.6 for *SparseNet* addresses all the concerns raised above. Fixed points of the map $\mathrm{S}_{\text{cw}}(\cdot)$ are explicitly characterized through some regularity conditions of the penalty functions as described in Section 2.6. Explicit convergence can be shown under additional conditions — Theorem 4.

Furthermore, a fixed point of the map $M_{\text{lla}}(\cdot)$ need not be a fixed point of $\mathrm{S}_{\text{cw}}(\cdot)$ – for specific examples see Section 2.2.3 in the secondary appendix. Every stationary point of the coordinate-wise procedure, however, is also a fixed point of MLLA.

The formal framework being laid, we proceed to perform some simple numerical experiments in Section 2.8.1. Further discussions and comparisons between coordinate-wise procedures and MLLA can be found in Section 2.2.3 in the secondary appendix.

2.8.1 Empirical performance and the role of calibration

In this section we show the importance of calibration and the specific manner in which the regularization path is traced out via *SparseNet* in terms of the quality of the solution obtained. In addition we compare the optimization performance of LLA/MLLA versus *SparseNet*.

In Figure 2.7 we consider an example with $n = p = 10$, $\Sigma = \{0.8^{|i-j|}\}1 \leq i, j, \leq p$, SNR=1, $\beta_1 = \beta_p = 1$, all other $\beta_j = 0$. We consider 50 values of γ and 20 values of λ, and compute the solutions for all methods at the recalibrated pairs $(\gamma, \lambda_S(\gamma, \lambda))$; see Figure 2.5 for the exact $\lambda_{k,\ell}$ values used in these plots.

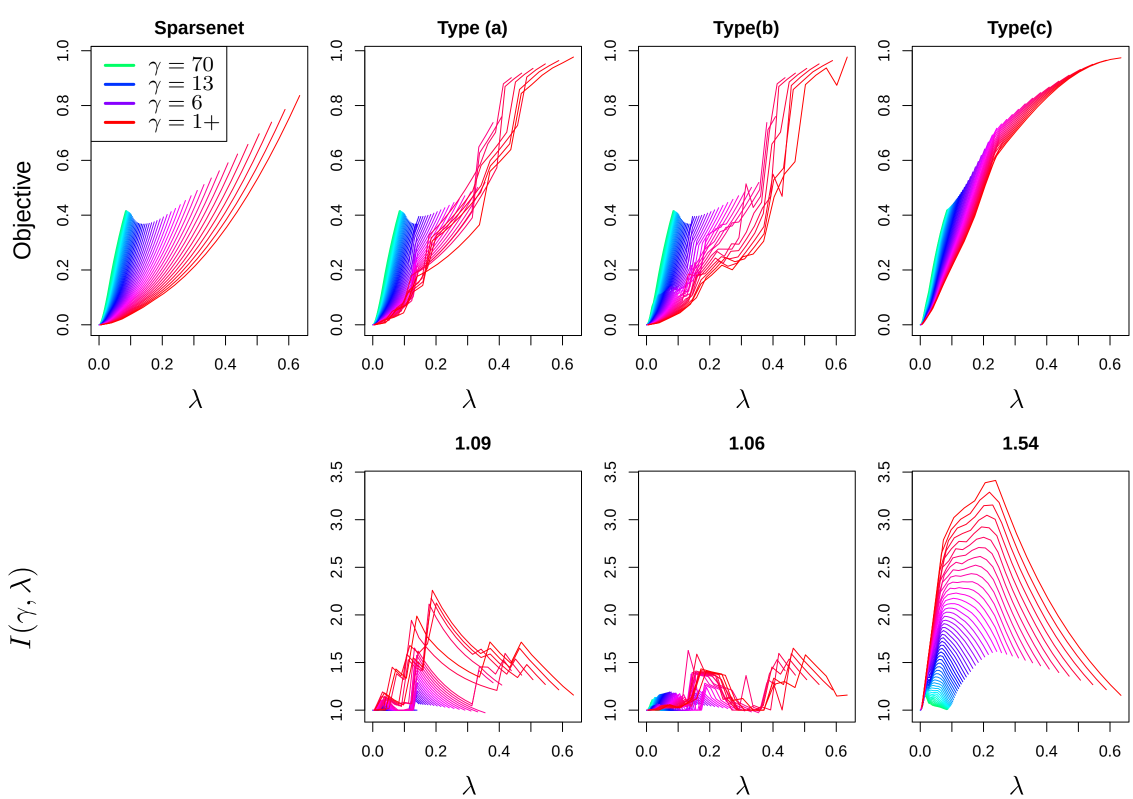

Figure 2.7: Top row: objective function for *SparseNet* compared to other coordinate-wise variants — Type (a), (b) and MLLA Type (c) — for a typical example. Plots are shown for 50 values of γ (some are labeled in the legend) and at each value of γ, 20 values of λ. Bottom row: relative increase $I(\gamma, \lambda)$ in the objective compared to *SparseNet*, with the average $\bar{I}$ reported at the top of each plot.

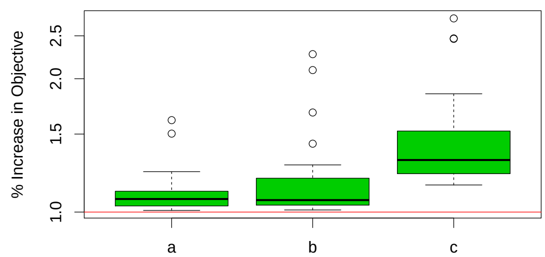

Figure 2.8: Boxplots of $\bar{I}$ over 24 different examples. In all cases *SparseNet* shows superior performance.

We compare *SparseNet* with

Type (a) Coordinate-wise procedure that computes solutions on a grid of λ for every fixed γ, with warm-starts across λ (from larger to smaller values). No *df* calibration is used here.

Type (b) Coordinate-wise procedure "cold-starting" with a zero vector (She 2009) for every pair (λ, γ).

Type (c) MLLA with the starting vector initialized to a vector of ones (Zhang 2009).

In all cases *SparseNet* shows superior performance. For each (γ, λ) pair we compute the relative increase $I(\gamma, \lambda) = [Q(\boldsymbol{\beta}^*) + 0.05]/[Q(\boldsymbol{\beta}^s) + 0.05]$, where $Q(\boldsymbol{\beta}^*)$ denotes the objective value obtained for procedure $*$ (types (a), (b) or (c) above), and $Q(\boldsymbol{\beta}^s)$ represents *SparseNet*. These values are shown in the second row of plots. They are also averaged over all values of (γ, λ) to give $\bar{I}$ (shown at the top of each plot). This is one of 24 different examples, over 6 different problem set-ups with 4 replicates in each. Figure 2.8 summarizes $\bar{I}$ for these 24 scenarios. Results are shown in Figure 2.7[4] The figure shows clearly the benefits of using the proposed calibration in terms of optimization performance. MLLA appears to get "stuck"

[4]We show the curves of the objective values, as in Figure 2.7, for all 24 scenarios in the secondary Appendix.

in local optima, supporting our claim in section 2.7.1. For larger values of γ the different coordinate-wise procedures perform similarly, since the problem is close to convex.

2.A Primary Appendix to Chapter 2

2.A.1 Convergence Analysis for Algorithm 1

To prove Theorem 4 we require Lemma 1.

Lemma 1. *Suppose the data* (y, X) *lies on a compact set. In addition, we assume the following*

1. *The penalty function* $P(t)$ *(symmetric around* 0*) is differentiable on* $t \geq 0$*;* $P'(|t|)$ *is non-negative, continuous and uniformly bounded, where* $P'(|t|)$ *is the derivative of* $P(|t|)$ *wrt* $|t|$.
2. *The sequence generated* $\{\boldsymbol{\beta}^k\}_k$ *is bounded*
3. *For every convergent subsequence* $\{\boldsymbol{\beta}^{n_k}\}_k \subset \{\boldsymbol{\beta}^k\}_k$ *the successive differences converge to zero:* $\boldsymbol{\beta}^{n_k} - \boldsymbol{\beta}^{n_k-1} \to 0$

If $\boldsymbol{\beta}^\infty$ *is any limit point of the sequence* $\{\boldsymbol{\beta}^k\}_k$*, then* $\boldsymbol{\beta}^\infty$ *is a minimum for the function* $Q(\boldsymbol{\beta})$*; i.e. for any* $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_p) \in \Re^p$

$$\liminf_{\alpha \downarrow 0+} \left\{ \frac{Q(\boldsymbol{\beta}^\infty + \alpha \boldsymbol{\delta}) - Q(\boldsymbol{\beta}^\infty)}{\alpha} \right\} \geq 0 \tag{2.35}$$

In the secondary Appendix 2.2 we present the proof of Lemma 1. We also show via Lemma 2 that assumption 2 is by no means restrictive. Lemma 3 gives a sufficient condition under which assumption 3 of Lemma 1 is true.

It is useful to note here that assumptions 2, 3 of Lemma 1 follow from the assumptions of Theorem 4 (we make a further note of this in the proof below).

We are now in a position to present the proof of Theorem 4:

Proof. **Proof of Theorem 4** We will assume WLOG that the data is standardized.

For fixed i and $(\beta_1, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_p)$, we will write $\chi^i_{(\beta_1,\ldots,\beta_{i-1},\beta_{i+1},\ldots,\beta_p)} \equiv \chi(u)$ for the sake notational convenience.

The sub-gradient(s) (Borwein & Lewis 2006) of $\chi(u)$ at u are given by

$$\partial\chi(u) = \nabla_i f((\beta_1, \ldots, \beta_{i-1}, u, \beta_{i+1}, \ldots, \beta_p)) + P'(|u|)\,\mathrm{sgn}(u) \tag{2.36}$$

Also observe that

$$\begin{aligned}
\chi(u+\delta) - \chi(u) =& \{f((\beta_1, \ldots, \beta_{i-1}, u+\delta, \beta_{i+1}, \ldots, \beta_p)) - \\
& f((\beta_1, \ldots, \beta_{i-1}, u, \beta_{i+1}, \ldots, \beta_p))\} + \{P(|u+\delta|) - P(|u|)\} \quad (2.37)\\
=& \{\nabla_i f((\beta_1, \ldots, \beta_{i-1}, u, \beta_{i+1}, \ldots, \beta_p))\delta + \tfrac{1}{2}\nabla_i^2 f \delta^2\}\\
&+ \{P'(|u|)(|u+\delta| - |u|) + \tfrac{1}{2}P''(|u^*|)(|u+\delta| - |u|)^2\} \quad (2.38)
\end{aligned}$$

Line (2.38) is obtained from (2.37) by a Taylor's series expansions on f (wrt to the i^{th} coordinate) and $|t| \mapsto P(|t|)$. $\nabla_i^2 f$ is the second derivative of the function f wrt the i^{th} coordinate and equals 1 as the features are all standardized; $|u^*|$ is some number between $|u+\delta|$ and $|u|$. The rhs of (2.38) can be simplified as follows:

$$\begin{aligned}
& \nabla_i f((\beta_1, \ldots, \beta_{i-1}, u, \beta_{i+1}, \ldots, \beta_p))\delta + \tfrac{1}{2}\nabla_i^2 f \delta^2\\
& + P'(|u|)(|u+\delta| - |u|) + \tfrac{1}{2}P''(|u^*|)(|u+\delta| - |u|)^2\\
= \; & \tfrac{1}{2}\nabla_i^2 f \delta^2 + \{\nabla_i f((\beta_1, \ldots, \beta_{i-1}, u, \beta_{i+1}, \ldots, \beta_p))\delta + P'(|u|)\,\mathrm{sgn}(u)\delta\} +\\
& \{P'(|u|)(|u+\delta| - |u|) - P'(|u|)\,\mathrm{sgn}(u)\delta\} + \tfrac{1}{2}P''(|u^*|)(|u+\delta| - |u|)^2
\end{aligned} \tag{2.39}$$

Since $\chi(u)$ is minimized at u_0; $0 \in \partial\chi(u_0)$. Thus using (2.36) we get

$$\nabla_i f((\beta_1, \ldots, \beta_{i-1}, u_0, \beta_{i+1}, \ldots, \beta_p))\delta + P'(|u_0|)\,\mathrm{sgn}(u_0)\delta = 0 \tag{2.40}$$

if $u_0 = 0$ then the above holds true for some value of $\text{sgn}(u_0) \in [-1, 1]$.

By definition of sub-gradient of $x \mapsto |x|$ and $P'(|x|) \geq 0$ we have

$$P'(|u|)(|u+\delta| - |u|) - P'(|u|)\,\text{sgn}(u)\delta = P'(|u|)\left((|u+\delta| - |u|) - \text{sgn}(u)\delta\right) \geq 0 \qquad (2.41)$$

Using (2.40,2.41) in (2.39,2.38) at $u = u_0$ we have

$$\chi(u_0+\delta) - \chi(u_0) \geq \tfrac{1}{2}\nabla_i^2 f \delta^2 + \tfrac{1}{2}P''(|u^*|)(|u_0+\delta| - |u_0|)^2 \qquad (2.42)$$

Observe that $(|u_0+\delta| - |u_0|)^2 \leq \delta^2$.

If $P''(|u^*|) \leq 0$ then $\frac{1}{2}P''(|u^*|)(|u_0+\delta| - |u_0|)^2 \geq \frac{1}{2}P''(|u^*|)\delta^2$.

If $P''(|u^*|) \geq 0$ then $\frac{1}{2}P''(|u^*|)(|u_0+\delta| - |u_0|)^2 \geq 0$.

Combining these two we get

$$\chi(u_0+\delta) - \chi(u_0) \geq \tfrac{1}{2}\delta^2\left(\nabla_i^2 f + \min\{P''(|u^*|), 0\}\right) \qquad (2.43)$$

By the conditions of this theorem, $(\nabla_i^2 f + \inf_x P''(|x|)) > 0$. Hence there exists a $\theta > 0$,

$$\theta = \tfrac{1}{2}\{\nabla_i^2 f + \min\{\inf_x P''(|x|), 0\}\} = \tfrac{1}{2}\{1 + \min\{\inf_x P''(|x|), 0\}\} > 0$$

which is independent of $(\beta_1, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_p)$, such that

$$\chi(u_0+\delta) - \chi(u_0) \geq \theta\delta^2 \qquad (2.44)$$

(For the MC+ penalty, $\inf_x P''(|x|) = -\frac{1}{\gamma}$; $\gamma > 1$.

For the SCAD, $\inf_x P''(|x|) = -\frac{1}{\gamma-1}$; $\gamma > 2$.)

In fact the dependence on u_0 above in (2.44) can be further relaxed. Following the above arguments carefully, we see that the function $\chi(u)$ is strictly convex (Borwein & Lewis 2006) as it satisfies:

$$\chi(u+\delta) - \chi(u) - \partial\chi(u)\delta \geq \theta\delta^2 \tag{2.45}$$

We will use this result to show the convergence of $\{\boldsymbol{\beta}^k\}$. Using (2.44) we have

$$\begin{aligned} Q(\boldsymbol{\beta}_i^{m-1}) - Q(\boldsymbol{\beta}_{i+1}^{m-1}) &\geq \theta(\beta_{i+1}^{m-1} - \beta_{i+1}^{m})^2 \\ &= \theta\|\boldsymbol{\beta}_i^{m-1} - \boldsymbol{\beta}_{i+1}^{m-1}\|_2^2 \end{aligned} \tag{2.46}$$

where $\boldsymbol{\beta}_i^{m-1} := (\beta_1^m, \dots, \beta_i^m, \beta_{i+1}^{m-1}, \dots, \beta_p^{m-1})$. (2.46) shows the boundedness of the sequence $\boldsymbol{\beta}^m$, for every $m > 1$ since the starting point $\boldsymbol{\beta}^1 \in \Re^p$.

Using (2.46) repeatedly across every coordinate, we have for every m

$$Q(\boldsymbol{\beta}^{m+1}) - Q(\boldsymbol{\beta}^m) \geq \theta\|\boldsymbol{\beta}^{m+1} - \boldsymbol{\beta}^m\|_2^2 \tag{2.47}$$

Using the fact that the (decreasing) sequence $Q(\boldsymbol{\beta}^m)$ converges, it is easy to see from (2.47), that the sequence $\boldsymbol{\beta}^k$ cannot cycle without convergence ie it must have a unique limit point. This completes the proof of convergence of $\boldsymbol{\beta}^k$.

Observe that, (2.47) implies that assumptions 2 and 3 of Lemma 1 are satisfied. Hence using Lemma 1, the limit of $\boldsymbol{\beta}^k$ is a minimum of $Q(\boldsymbol{\beta})$ — this completes the proof. □

2.2 Secondary Appendix to Chapter 2

We will first address the boundedness of the sequence $\boldsymbol{\beta}^k$. For this purpose we present the following Lemma:

Lemma 2. *Suppose the data $(y, \mathbf{X})$ lies on a compact set. For fixed $P(\cdot;\lambda;\gamma)$ and $\boldsymbol{\beta}_0 \in \Re^p$ let $\mathcal{H}_\prime = \{\boldsymbol{\beta} | Q(\boldsymbol{\beta}) \leq Q(\boldsymbol{\beta}_0)\}$. There exists a bounded set $\tilde{\mathcal{H}}_0 \subset \Re^p$ such that*

$$\{Q(\boldsymbol{\beta}) | \boldsymbol{\beta} \in \mathcal{H}_\prime\} = \{Q(\boldsymbol{\beta}) | \boldsymbol{\beta} \in \tilde{\mathcal{H}}_0\} \tag{2.48}$$

Proof. This lemma is trivially true if $P(|t|;\lambda;\gamma)$ is unbounded in $|t|$.

The lemma requires a proof if $P(|t|;\lambda;\gamma)$ is uniformly bounded in $|t|$ (MC+ penalty, for example). We prove this part via contradiction.

Suppose there exists a sequence, $\{\boldsymbol{\beta}^k\}_k \subset \mathcal{H}\prime$ such that $\|\boldsymbol{\beta}^k\|_2 \to \infty$. This necessarily implies that, there is a subset of indices $(i_1,\ldots,i_m) \subset \{1,\ldots,p\}$ such that $\{(\beta^k_{i_1},\ldots,\beta^k_{i_m})\}_k$ is an unbounded sequence. This sequence satisfies (passing on to a subsequence if necessary),

$$\mathbf{X}[i_1,\ldots,i_m](\beta^k_{i_1},\ldots,\beta^k_{i_m})' = \mathbf{X}[i_1,\ldots,i_m](\beta^0_{i_1},\ldots,\beta^0_{i_m})'$$

for some $(\beta^0_{i_1},\ldots,\beta^0_{i_m}) \in \Re^m$ independent of k and $\mathbf{X}[i_1,\ldots,i_m] = [\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_m}]$.

The sequence $\{\boldsymbol{\beta}^k\}_k \subset \mathcal{H}\prime$, considered above is arbitrary. Hence the range of values taken by the function $Q(\boldsymbol{\beta})$ remains the same if one restricts to bounded subsets of $\Re^p$. So there exists a bounded set $\tilde{\mathcal{H}}_0 \subset \Re^p$ such that (2.48) holds true. $\square$

Lemma 2, shows that the boundedness assumption on the coordinate-wise updates is not by any means restrictive. If required, one can restrict $\|\boldsymbol{\beta}\|_\infty \leq M$, in the Algorithm for some $M > 0$, sufficiently large.

Lemmas 1 and 3 characterize the fate of *limit points* of the sequence produced by the coordinate-wise updates under milder conditions than those required by Theorem 4.

Proof of Lemma 1

Proof. For any $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_p)$ and $\boldsymbol{\delta}_i = (0,\ldots,\delta_i,\ldots,0) \in \Re^p$,

$$\liminf_{\alpha\downarrow 0+}\left\{\frac{Q(\boldsymbol{\beta}+\alpha\boldsymbol{\delta}_i) - Q(\boldsymbol{\beta})}{\alpha}\right\} = \nabla_i f(\boldsymbol{\beta})\delta_i + \liminf_{\alpha\downarrow 0+}\left\{\frac{P(|\beta_i+\alpha\delta_i|) - P(|\beta_i|)}{\alpha}\right\} \tag{2.49}$$

where $f(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$, ∇_i is the partial derivative w.r.t. the i^{th} coordinate.

Observe that the second term in (2.49) simplifies to

$$\begin{aligned}\partial P(\beta_i;\delta_i) &:= \liminf_{\alpha\downarrow 0+}\left\{\frac{P(|\beta_i+\alpha\delta_i|)-P(|\beta_i|)}{\alpha}\right\}\\ &= \begin{cases} P'(|\beta_i|)\,\mathrm{sgn}(\beta_i)\delta_i & \text{if } |\beta_i|>0;\\ P'(0)|\delta_i| & \text{if } |\beta_i|=0.\end{cases}\end{aligned} \tag{2.50}$$

where

$$\mathrm{sgn}(x)\in\begin{cases}\{1\} & \text{if } x>0;\\ \{-1\} & \text{if } x<0;\\ \text{[-1,1]} & \text{if } x=0.\end{cases}$$

Assume $\boldsymbol{\beta}^{n_k}\to\boldsymbol{\beta}^{\infty}=(\beta_1^{\infty},\ldots,\beta_p^{\infty})$. Using (2.50) and Assumption 3, as $k\to\infty$

$$\boldsymbol{\beta}_i^{n_k-1} := (\beta_1^{n_k},\ldots,\beta_{i-1}^{n_k},\beta_i^{n_k},\beta_{i+1}^{n_k-1},\beta_p^{n_k-1})\to(\beta_1^{\infty},\ldots,\beta_{i-1}^{\infty},\beta_i^{\infty},\beta_{i+1}^{\infty},\beta_p^{\infty}) \tag{2.51}$$

$$\text{If, }\ \beta_i^{\infty}\neq 0, \partial P(\beta_i^{n_k};\delta_i)\to\partial P(\beta_i^{\infty};\delta_i);\quad \text{If, }\ \beta_i^{\infty}=0, \partial P(\beta_i^{\infty};\delta_i)\geq\liminf_k \partial P(\beta_i^{n_k};\delta_i)$$

By definition of a coordinate-wise minimum, using (2.49,2.50) we have

$$\nabla_i f(\boldsymbol{\beta}_i^{n_k-1})\delta_i+\partial P(\beta_i^{n_k};\delta_i)\geq 0\ \forall k \tag{2.52}$$

The above imply $\forall i\in\{1,\ldots,p\}$

$$\nabla_i f(\boldsymbol{\beta}^{\infty})\delta_i+\partial P(\beta_i^{\infty};\delta_i)\geq\liminf_k\left\{\nabla_i f(\boldsymbol{\beta}_i^{n_k-1})\delta_i+\partial P(\beta_i^{n_k};\delta_i)\right\}\underbrace{\geq}_{\text{by (2.52)}} 0 \tag{2.53}$$

Using (2.49,2.50) the l.h.s. of (2.53) gives

$$\liminf_{\alpha\downarrow 0+}\left\{\frac{Q(\boldsymbol{\beta}^{\infty}+\alpha\boldsymbol{\delta}_i^{\infty})-Q(\boldsymbol{\beta}^{\infty})}{\alpha}\right\}\geq 0 \tag{2.54}$$

For $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_p) \in \Re^p$, due to differentiability of $f(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$

$$\begin{aligned}
\liminf_{\alpha \downarrow 0+} \left\{ \frac{Q(\boldsymbol{\beta}^\infty + \alpha\boldsymbol{\delta}) - Q(\boldsymbol{\beta}^\infty)}{\alpha} \right\} &= \sum_{j=1}^p \nabla_j f(\boldsymbol{\beta}^\infty)\delta_j + \sum_{j=1}^p \lim_{\alpha\downarrow 0+} \frac{P(|\beta_j^\infty + \alpha\delta_j|) - P(|\beta_j^\infty|)}{\alpha} \\
&= \sum_{j=1}^p \left\{ f(\boldsymbol{\beta}^\infty)\delta_j + \lim_{\alpha\downarrow 0+} \frac{P(|\beta_j^\infty + \alpha\delta_j|) - P(|\beta_j^\infty|)}{\alpha} \right\} \\
&= \sum_{j=1}^p \ \liminf_{\alpha\downarrow 0+} \left\{ \frac{Q(\boldsymbol{\beta}^\infty + \alpha\boldsymbol{\delta}_j^\infty) - Q(\boldsymbol{\beta}^\infty)}{\alpha} \right\} \underbrace{\geq}_{\text{By (2.54)}} 0
\end{aligned} \tag{2.55}$$

This completes the proof. □

Lemma 3 gives a sufficient condition under which Assumption 3 of Lemma 1 is true.

Lemma 3. *Suppose in addition to Assumption 1 of Lemma 1 the following assumptions hold:*

1. *For every $i = 1, \ldots, p$ and $(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p) \in \Re^{p-1}$ the univariate function*

$$\chi^i_{(u_1,\ldots,u_{i-1},u_{i+1},\ldots,u_p)}(\cdot) : u \mapsto \quad Q(u_1, \ldots, u_{i-1}, u, u_{i+1}, \ldots, u_p) \tag{2.56}$$

has a unique global minimum.

2. *If $u_G \equiv u_G(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p)$ is the global minimum and $u_L \equiv u_L(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p)$ any other local minimum of the function (2.56); there exists $\epsilon > 0$, independent of the choice of $(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p)$ such that*

$$|u_G(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p) - u_L(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_p)| \geq \epsilon$$

— that is u_G and u_L are uniformly separated.

Then for every convergent subsequence $\{\boldsymbol{\beta}^{n_k}\}_k \subset \{\boldsymbol{\beta}^k\}_k$, the successive differences $\boldsymbol{\beta}^{n_k} - \boldsymbol{\beta}^{n_k - 1} \to 0$; i.e. Assumption 3 of Lemma 1 is true.

Proof. Borrowing notation from (2.51), consider the sequence $\{\boldsymbol{\beta}_{p-1}^{n_k-1}\}$

$$\boldsymbol{\beta}_{p-1}^{n_k-1} := (\beta_1^{n_k}, \ldots, \beta_{p-2}^{n_k}, \beta_{p-1}^{n_k}, \beta_p^{n_k-1}) \tag{2.57}$$

where $\boldsymbol{\beta}_{p-1}^{n_k-1}$ differs from $\boldsymbol{\beta}^{n_k}$ at the p^{th} coordinate. We will show $\boldsymbol{\beta}_{p-1}^{n_k-1} - \boldsymbol{\beta}^{n_k} \to 0$ by the method of contradiction.

Writing $\boldsymbol{\beta}^{n_k} - \boldsymbol{\beta}_{p-1}^{n_k-1} = \boldsymbol{\Delta}_p^{n_k} := (0, \ldots, 0, \Delta_p^{n_k})$ we have

$$Q(\boldsymbol{\beta}^{n_k} - \boldsymbol{\Delta}_p^{n_k}) - Q(\boldsymbol{\beta}^{n_k}) \downarrow 0 \ \text{ as } n_k \to \infty$$

As $k \to \infty$, let $\boldsymbol{\beta}^{n_k} \to \boldsymbol{\beta}^+$. Passing on to a subsequence (if necessary) we have $\Delta_p^{n_k} \to \Delta_p^+ \neq 0$ (as $\Delta_p^{n_k} \nrightarrow \mathbf{0}$) and

$$\boldsymbol{\beta}^{n_k} - \boldsymbol{\Delta}_p^{n_k} \to \boldsymbol{\beta}^+ - \boldsymbol{\Delta}_p^+$$

Furthermore, the continuity of the function $Q(\cdot)$ implies

$$\big(Q(\boldsymbol{\beta}^{n_k} - \boldsymbol{\Delta}_p^{n_k}) - Q(\boldsymbol{\beta}^{n_k})\big) \downarrow \big(Q(\boldsymbol{\beta}^+ - \boldsymbol{\Delta}_p^+) - Q(\boldsymbol{\beta}^+)\big) = 0 \tag{2.58}$$

By the definition of $\boldsymbol{\beta}^{n_k} := (\beta_1^{n_k}, \ldots, \beta_p^{n_k})$ we have

$$\nabla_p f(\boldsymbol{\beta}^{n_k})\delta_p + \partial P(\beta_p^{n_k}; \delta_p) \geq 0, \quad \forall k$$

Passing on to the limit $k \to \infty$ we get (using arguments as in proof of Lemma 1)

$$\nabla_p f(\boldsymbol{\beta}^+)\delta_p + \partial P(\beta_p^+; \delta_p) \geq 0$$

where $\boldsymbol{\beta}^+ = (\beta_1^+, \ldots, \beta_p^+)$. This implies that $\chi^p_{(\beta_1^+, \ldots, \beta_{p-1}^+)}(u)$ has *a minimum* at $u = \beta_p^+$. By Assumption 2 of this Lemma the global minimum u_G is uniformly separated from other local minima u_L — hence β_p^+ is the *global* minimum. But (2.58) implies that $\chi^p_{(\beta_1^+, \ldots, \beta_{p-1}^+)}(u)$ has two distinct global minima at $u = \beta_p^+$ and $u = \beta_p^+ - \Delta_p^+$ — a contradiction to Assumption 1. Hence $\Delta_p^+ = 0$ and $\beta_p^{n_k-1} - \beta_p^{n_k} \to 0$.

In the above spirit consider sequence (2.59) (differing from $\boldsymbol{\beta}_{p-1}^{n_k-1}$ at $(p-1)^{\text{th}}$ coordinate)

$$\boldsymbol{\beta}_{p-2}^{n_k-1} := (\beta_1^{n_k}, \ldots, \beta_{p-3}^{n_k}, \beta_{p-2}^{n_k}, \beta_{p-1}^{n_k-1}, \beta_p^{n_k-1}) \tag{2.59}$$

We will show $\boldsymbol{\beta}_{p-2}^{n_k-1} - \boldsymbol{\beta}_{p-1}^{n_k-1} \to 0$, which along with $\boldsymbol{\beta}_{p-1}^{n_k-1} - \boldsymbol{\beta}^{n_k} \to 0$ (shown above) will imply $\boldsymbol{\beta}_{p-2}^{n_k-1} - \boldsymbol{\beta}^{n_k} \to 0$. The proof of this part is similar to the above.

Write $\boldsymbol{\beta}_{p-1}^{n_k-1} - \boldsymbol{\beta}_{p-2}^{n_k-1} = \boldsymbol{\Delta}_{p-1}^{n_k-1} := (0, \ldots, 0, \Delta_{p-1}^{n_k}, 0)$ and suppose $\Delta_{p-1}^{n_k} \nrightarrow 0$. We will arrive at a contradiction.

By hypothesis, there is a subsequence of $\Delta_{p-1}^{n_k}$ converging to $\Delta_{p-1}^+ \neq 0$. Passing onto a further subsequence, if necessary and using $\boldsymbol{\Delta}_{p-1}^{n_k-1} \to \boldsymbol{\Delta}_{p-1}^+$ we have

$$\Big(Q(\boldsymbol{\beta}_{p-1}^{n_k-1} - \boldsymbol{\Delta}_{p-1}^{n_k-1}) - Q(\boldsymbol{\beta}_{p-1}^{n_k-1})\Big) \downarrow \Big(Q(\boldsymbol{\beta}^+ - \boldsymbol{\Delta}_{p-1}^+) - Q(\boldsymbol{\beta}^+)\Big) = 0 \tag{2.60}$$

This essentially implies that the map $\chi_{(\beta_1^+,\ldots,\beta_{p-2}^+,\beta_p^+)}^{p-1}(u)$ has two distinct global minima at $u = \beta_{p-1}^+$ and $u = \beta_{p-1}^+ - \Delta_{p-1}^+$; which is a contradiction — hence $\Delta_{p-1}^{n_k} \to 0$.

We can follow the above arguments inductively for finitely many steps namely $i = p-3, p-4, \ldots, 1, p$ and show that $\Delta_i^{n_k} \to 0$ for every i. This shows $\boldsymbol{\beta}^{n_k-1} - \boldsymbol{\beta}^{n_k} \to 0$, thereby completing the proof. □

The uniform separation condition — Assumption 2 of Lemma 3 is a trivial consequence of strict convexity of the univariate functions (Theorem 4) and are more general.

2.2.1 Calibration of MC+ and Monotonicity Properties of $\lambda_S, \gamma\lambda_S$

Proof of Theorem 2 on page 25

Proof. For convenience we rewrite $\lambda_S(\lambda, \gamma)$ in terms of a function $f_\lambda(\gamma)$ as

$$\lambda_S(\lambda, \gamma) = \lambda f_\lambda(\gamma). \tag{2.61}$$

Using (2.28) and the equality of effective *df* across all γ for every λ, $f_\lambda(\gamma)$ can be defined as a solution to the following functional equation

$$\frac{\gamma}{\gamma - 1}\Pr(\lambda f_\lambda(\gamma) \leq |\tilde{\beta}| < \gamma\lambda f_\lambda(\gamma)) \quad + \quad \Pr(|\tilde{\beta}| > \gamma\lambda f_\lambda(\gamma)) = \Pr(|\tilde{\beta}| > \lambda) \tag{2.62}$$

$$\forall\gamma > 1,\ f_\lambda(\gamma) > 1. \tag{2.63}$$

Since we have assumed $\sigma^2/n = 1$ and a null generative model, we have

$$\frac{\gamma}{\gamma - 1}(\Phi(\gamma\lambda f_\lambda(\gamma)) - \Phi(\lambda f_\lambda(\gamma)) + (1 - \Phi(\gamma\lambda f_\lambda(\gamma)) = (1 - \Phi(\lambda)) \tag{2.64}$$

or equivalently

$$\Phi(\gamma\lambda f_\lambda(\gamma)) - \gamma\Phi(\lambda f_\lambda(\gamma)) = -(\gamma - 1)\Phi(\lambda) \tag{2.65}$$

where Φ is the standard normal cdf and ϕ the pdf of the same. Equation (2.29) is obtained from (2.65) and (2.61). $\square$

Proof of Theorem 3 on page 25 Before we begin the proof, we need a lemma.

Lemma 4. *For $\delta \leq \lambda_H$, where $\lambda_H = \lambda_S(\lambda, \gamma = 1+)$ the function $h(\delta)$*

$$h(\delta) = \Phi(\delta) - \Phi(\lambda) - \delta\phi(\delta) \tag{2.66}$$

satisfies $h(\delta) < 0,\ \ \forall\lambda_H > \delta > \lambda$

Proof. Observe that for every fixed $\lambda > 0$, $h(\lambda) = -\delta\phi(\delta) < 0$. In addition, the derivative of $h(\cdot)$ satisfies

$$h'(\delta) = -\delta\phi'(\delta) > 0 \tag{2.67}$$

since $\phi(\delta)$ is the standard normal density and $\delta > 0$.

Also, from equation (2.64) it follows that (taking $\gamma \to 1+$),

$$\lambda_H \phi(\lambda_H) + 1 - \Phi(\lambda_H) = 1 - \Phi(\lambda)$$

which implies $\Phi(\lambda_H) - \Phi(\lambda) - \lambda_H \phi(\lambda_H) = 0$ that is $h(\lambda_H) = 0$. Since $h(\cdot)$ is strictly monotone increasing on $[\lambda, \lambda_H]$, with $h(\lambda) < 0$ and $h(\lambda_H) = 0$ it is necessarily negative on $[\lambda, \lambda_H)$. This completes the proof. □

Proof. Proof of part [a]. Let $\lambda'_{\max}$ denote the partial derivative of $\lambda_{\max} \equiv \gamma\lambda_S(\lambda, \gamma)$ w.r.t γ.

Differentiating both sides of equation (2.29) w.r.t γ, we get

$$\lambda'_{\max}(\phi(\gamma\lambda_S) - \phi(\lambda_S)) = \Phi(\lambda_S) - \Phi(\lambda) - \lambda_S \phi(\lambda_S) \tag{2.68}$$

Observe that $\phi(\gamma\lambda_S) < \phi(\lambda_S)$, since $\gamma\lambda_S > \lambda_S$.

In order to complete the proof it suffices to show that $\Phi(\lambda_S) - \Phi(\lambda) - \lambda_S \phi(\lambda_S) < 0$. This follows from Lemma 4, which gives $h(\lambda_S) < 0$ since $\lambda < \lambda_S < \lambda_H$.

Proof of part [b]. Let $\lambda'_{\min}$ denote the partial derivative of $\lambda_{\min} \equiv \lambda_S(\lambda, \gamma)$ w.r.t γ. Differentiating both sides of equation (2.29) w.r.t γ, we get

$$\lambda'_{\min}\gamma(\phi(\gamma\lambda_S) - \phi(\lambda_S)) = \Phi(\lambda_S) - \Phi(\lambda) - \lambda_S \phi(\gamma\lambda_S) \tag{2.69}$$

Since $\phi(\gamma\lambda_S) < \phi(\lambda_S)$, it suffices to show that r.h.s of (2.69) is greater than zero. Define for a fixed $\lambda > 0$ the function

$$g(\lambda_S, \gamma) = \Phi(\lambda_S) - \phi(\gamma\lambda_S) \tag{2.70}$$

From (2.29) it follows that,

$$\Phi(\lambda_S) = \frac{\gamma - 1}{\gamma}\Phi(\lambda) + \Phi(\lambda_S\gamma)/\gamma \tag{2.71}$$

Using (2.71), $g(\lambda_S, \gamma)$ becomes

$$g(\lambda_S, \gamma) = \Phi(\lambda) - \frac{\Phi(\lambda)}{\gamma} + \frac{\Phi(\lambda_S\gamma)}{\gamma} - \phi(\lambda_S\gamma)\lambda_S \tag{2.72}$$

Consider the function $g(\lambda_S, \gamma) - \Phi(\lambda)$, for fixed $\lambda > 0$.

By Theorem 3 (part [a]), we know that for fixed λ, the map $\gamma \mapsto \lambda_{\max} = \gamma\lambda_S$ is increasing in γ. Hence, the function $g(\lambda_S, \gamma) - \Phi(\lambda)$ is greater than zero iff the function $h(u)$ (with $u = \gamma\lambda_S$)

$$h(u) = \Phi(u) - u\phi(u) - \Phi(\lambda)$$

satisfies

$$h(u) > 0 \quad \text{on } u > \lambda_H$$

Observe that $h'(u) = -u\phi'(u) > 0$ on $u > \lambda_H$. In addition, from (2.64), taking limits over γ it follows that $h(\lambda_H) = 0$.

This proves that $h(u) > 0$ on $u > \lambda_H$. Consequently, $g(\lambda_S, \gamma) > \Phi(\lambda) \quad \forall\gamma > 1$. This proves that the r.h.s of (2.69) is greater than zero. □

2.2.2 Parametric functional form for $\lambda_S(\lambda, \gamma)$

We detail out the steps leading to the expressions (2.30,2.31) given in Section 2.5.1.

For every fixed $\lambda > 0$ the following are the list of properties that the function $f_\lambda(\gamma)$ (see Appendix 2.2.1) should have

1. monotone increase of shrinkage thresholds λ_S from soft to hard thresholding, implies $\lambda f_\lambda(\gamma)$ should be increasing as γ decreases.

2. monotone decrease of bias thresholds $\gamma\lambda_S$ from soft to hard thresholding requires $\gamma\lambda f_\lambda(\gamma)$ to be increasing as γ increases.

3. The sandwiching property:

$$\lambda_S(\gamma) > \lambda \; \forall \gamma \in (1,\infty) \implies f_\lambda(\gamma) > 1 \quad \forall \gamma \in (1,\infty) \tag{2.73}$$

4. The *df* calibration at the two extremes of the family: $df(\hat{\boldsymbol{\mu}}_{\gamma=1+,\lambda}) = df(\hat{\boldsymbol{\mu}}_{\gamma=\infty,\lambda})$.

The above considerations suggest a nice parametric form for $f_\lambda(\gamma)$ given by

$$f_\lambda(\gamma) = 1 + \tau/\gamma \quad \text{for some} \quad \tau > 0 \tag{2.74}$$

Let us denote $\lambda_S(\lambda, 1+) = \lambda_H$

Equating the *df*'s for the soft and hard thresholding τ is to be chosen as:

$$(1 - \Phi(\lambda)) = \lambda_H \phi(\lambda_H) + (1 - \Phi(\lambda_H))$$

which implies

$$(1+\tau)^{-1} = f_\lambda(\gamma = 1+) = \lambda_H^{-1} \Phi^{-1}(\Phi(\lambda_H) - \lambda_H \phi(\lambda_H))$$

Hence we get

$$\lambda_S = \lambda_H \left(\frac{1}{1+\tau} + \frac{\tau}{\gamma(1+\tau)} \right)$$

In addition, observe that as $\gamma \to \infty$, $\lambda_S(\lambda,\gamma) \to \frac{\lambda_H}{1+\tau} = \lambda$. This gives expression (2.31).

Tightening up the initial approximation of λ_S

In order to get a better approximation of the mapping $(\lambda,\gamma) \mapsto (\lambda_S, \gamma\lambda_S)$, we can define a sequence of recursive updates based on equation (2.29). Theorem 5 gives a proof of the convergence of these updates and the algorithm for obtaining the better approximations are given in Section (2.2.2).

Theorem 5. *With reference to (2.29), define a sequence of estimates* $\{z_k\}_{k\geq 0}$ *recursively as*

$$\frac{1}{\gamma-1}\Phi(\gamma z_k) - \frac{\gamma}{\gamma-1}\Phi(z_{k+1}) = -\Phi(\lambda) \tag{2.75}$$

where the variables $\{z_k\}_{k\geq 0}$ *are actually the iterative updates of* $\lambda_S(\gamma)$ *for a fixed* (λ,γ). *Under the condition*

$$\sup_k\{\frac{|\phi(\xi_k^*)|}{|\phi(\xi_k)|}\} < 1 \tag{2.76}$$

where $\xi_k^* \in (\min\{\gamma z_k, \gamma z_{k+1}\}, \max\{\gamma z_k, \gamma z_{k+1}\})$ *and* $\xi_k \in (\min\{z_{k+1}, z_{k+2}\}, \max\{z_{k+1}, z_{k+2}\})$ *for all* $k \geq 0$, *the sequence* $\{z_k\}_{k\geq 0}$ *converges to a fixed point* z^* *of the following equation:*

$$\frac{1}{\gamma-1}\Phi(\gamma z^*) - \frac{\gamma}{\gamma-1}\Phi(z^*) = -\Phi(\lambda) \tag{2.77}$$

and hence provides a solution to $(\lambda,\gamma) \mapsto (\lambda_S(\gamma) = z^*, \gamma\lambda_S(\gamma) = \gamma z^*)$

Proof. By the definition of the updates z_k, in (2.75) we get for $k = k, k+1$

$$\frac{1}{\gamma-1}\Phi(\gamma z_k) - \frac{\gamma}{\gamma-1}\Phi(z_{k+1}) = -\Phi(\lambda) \quad k \in \{k, k+1\}$$

Using the above two we get:

$$\Phi(\gamma z_k) - \Phi(\gamma z_{k+1}) = \gamma(\Phi(z_{k+1}) - \Phi(z_{k+2})) \tag{2.78}$$

Applying the mean-value theorem on both sides of (2.78) we get:

$$(\gamma z_k - \gamma z_{k+1})\phi(\gamma\eta_k) = \gamma(z_{k+1} - z_{k+2})\phi(\eta_k^*) \tag{2.79}$$

where η_k lies in between z_k,z_{k+1} and η_k^* lies in between z_{k+1},z_{k+2}.

Writing $R_k = \frac{\phi(\gamma\eta_k)}{\phi(\eta_k^*)}, \ \forall k \geq 0$, we observe that

$$|z_{m+1} - z_{m+2}| < C(\sup_k |R_k|)^m$$

for some C which does not depend upon m. Suppose $\sup_k |R_k| < 1$, then we observe that the telescopic sum $\sum_k (z_{k+1} - z_k)$ converges absolutely. Hence in particular the sequence z_k converges.

The condition (2.76) stated in this theorem, is a sufficient condition for $\sup_k |R_k| < 1$, hence the sequence of updates converge. Observe that, the sufficient condition is very plausible since $\gamma > 1$, and the density $\phi(\cdot)$ decreases on the positive reals.

This completes the proof. □

Algorithm for tightening up the approximation of *shrinkage thresholds*

For a given (λ, γ) the re-parametrized MC+ penalty is of the form given in (2.26) where $\lambda_S, \gamma\lambda_S$ defined through (2.29) are obtained by the following steps:

1. Obtain $f_\lambda(\gamma)$ based on the starting proposal function as in (2.75). This gives an estimate for $\hat{\lambda}_{\min}$ and $\hat{\lambda}_{\max} = \gamma\lambda_S$ using (2.61).

2. With $z_0 = \hat{\lambda}_{\min}$ obtain recursively the sequence of estimates $\{z_k\}_{k \geq 1}$ using Theorem (5) till convergence criterion is met, that is, $|df(\hat{\boldsymbol{\mu}}_{\gamma,\lambda}) - df(\hat{\mu}_{\gamma=1+}(y, \lambda))| < \text{tol}$.

3. If the sequence $z_k \to z_*$ then assign $\lambda_S = z_*$ and $\gamma\lambda_S = \gamma z_*$

Theorem 5 gives a formal proof of the convergence of the above algorithm.

2.2.3 Understanding the sub-optimality of LLA through examples

In this section, as a continuation to Section 2.8 we explore the reasons behind the sub-optimality of LLA through two univariate problem instances.

Example 1: log-penalty and the phase-transition phenomenon

Consider the log-penalized least-squares criterion (2.10)

$$Q^{(1)}(\beta) = \tfrac{1}{2}(\beta - \tilde{\beta})^2 + \frac{\lambda}{\log(\gamma + 1)} \log(\gamma|\beta| + 1) \tag{2.80}$$

for some large value of γ such that $Q^{(1)}(\beta)$ has multiple local minima, but a unique global minimizer. The corresponding (multi-stage) LLA for this problem leads to a sequence $\{\beta^k\}_{k\geq 1}$ defined by

$$\begin{aligned} \beta^{k+1} &= \arg\min_{\beta} \quad \tfrac{1}{2}(\beta - \tilde{\beta})^2 + w^k|\beta| \\ &= \mathrm{S}(\tilde{\beta}, w^k), \quad \text{with } \; w^k = \frac{\lambda\gamma}{(\gamma|\beta^k| + 1)\log(\gamma + 1)} \end{aligned} \tag{2.81}$$

Convergence of the sequence of estimates β^k corresponds to fixed points of the function

$$M_{\text{lla}} \; : \; \beta \mapsto \operatorname{sgn}(\tilde{\beta})(|\tilde{\beta}| - \frac{\lambda^*}{(\gamma|\beta| + 1)})_+ \quad \text{with} \quad \lambda^* = \frac{\lambda\gamma}{\log(\gamma + 1)}. \tag{2.82}$$

$M_{\text{lla}}(\beta)$ on $\beta \geq 0$, has multiple fixed points for certain choices of $\lambda, \tilde{\beta}$.

For example, let $\gamma = 500$, $\lambda = 74.6$ and $\tilde{\beta} = 10$; here $Q(\beta)$, $\beta \geq 0$ has two minima, with the global minimum at zero. Figure 2.9[Right] shows the iteration map $M_{\text{lla}}(\beta)$. It has two fixed points, corresponding to the two local minima $0 = \beta_1 < \beta_2$ of $Q(\beta)$, $\beta \geq 0$. There exists a critical value $r > 0$ such that

$$\beta^1 < r \Longrightarrow \beta^k \to 0 = \beta_1, \quad \beta^1 > r \Longrightarrow \beta^k \to \beta_2 \;\text{ as } k \to \infty \tag{2.83}$$

This *phase transition* phenomenon is responsible for the sub-optimality of the LLA in the one-dimensional case. Note that the coordinate-wise procedure applied in this context will give the global minimum.

The importance of the initial choice in obtaining "optimal" estimates via the re-weighted ℓ_1 minimization has also been pointed out in an *asymptotic* setting by Huang et al. (2008); our observations are non-asymptotic.

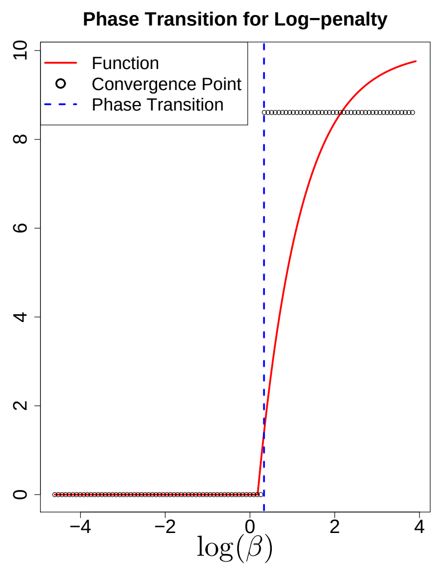

Figure 2.9: $M_{\text{lla}}(\beta)$ for the univariate log-penalized least squares problem. The starting point β^1 determines which stationary point the LLA sequence will converge to. If the starting point is on the right side of the phase transition line (vertical dotted line), then the LLA will converge to a strictly sub-optimal value of the objective function. In this example, the global minimum of the one-dimensional criterion is at zero.

Example 2: MC+ penalty and rate of convergence

Here we study the rate of convergence to the global minimum for the MC+ penalized least squares problem (2.16). The multi-stage LLA for this problem gives the following updates

$$\begin{aligned} \beta^{k+1} &= \arg\min_{\beta} \quad \tfrac{1}{2}(\beta - \tilde{\beta})^2 + w^k|\beta| \\ &= \mathrm{S}(\tilde{\beta}, w^k), \quad \text{with } \; w^k = \lambda(1 - \frac{|\beta^k|}{\gamma\lambda})_+ \end{aligned} \tag{2.84}$$

The function $\tilde{\beta} \mapsto \mathrm{S}(\tilde{\beta}, \lambda(1 - \frac{|\beta^k|}{\gamma\lambda})_+)$ need not be a contraction mapping, hence one cannot readily appeal to Banach's Fixed Point Theorem to obtain a proof of the convergence and a bound on its rate. However, in this example we can justify convergence (see Lemma 5)

Lemma 5. *Given $\tilde{\beta}, \gamma > 1, \lambda > 0$ the sequence $\{\beta^{k+1}\}_k$ defined in (2.84) converges to the minimum of $Q^{(1)}(\beta)$ for the MC+ penalized criterion at a (worst-case) geometric rate*

$$|\beta^k - \arg\min Q^{(1)}(\beta)| = O(1/\gamma^k)$$

Sketch of proof. Assume, for the sake of simplicity $\tilde{\beta} \geq 0, \beta^k \geq 0$. The update β^{k+1} is given explicitly by

$$\beta^{k+1} = \begin{cases} \tilde{\beta} - \lambda + \frac{1}{\gamma}\beta^k & \text{if } \gamma(\lambda - \tilde{\beta}) \leq \beta^k \leq \lambda\gamma \\ 0 & \text{if } \gamma(\lambda - \tilde{\beta}) > \beta^k \leq \lambda\gamma \\ \tilde{\beta} & \text{if } \beta^k > \lambda\gamma \end{cases}$$

On iteratively expanding a particular case of the above expression, we obtain

$$\beta^{k+1} = (\tilde{\beta} - \lambda)(\sum_{i=0}^{k} \gamma^{-i}) + \gamma^{-k-1}\beta^1, \text{ if } \gamma(\lambda - \tilde{\beta}) \leq \beta^i \leq \lambda\gamma \ \forall, 1 \leq i \leq k \tag{2.85}$$

The proof essentially considers different cases based on where β^k lies wrt $\lambda\gamma, \gamma(\lambda - \tilde{\beta})$ and derives expressions for the sequence β^k. The convergence of β^k follows in a fairly straightforward way analyzing these expressions. We omit the details for the sake of brevity.

The following corollary is a consequence of Lemma 5.

Corollary 2. *The number of iterations required for the multi-stage LLA algorithm of the univariate MC+ penalized least squares to converge within an ϵ tolerance of the minimizer of the objective $Q^{(1)}(\boldsymbol{\beta})$ is of the order of $-\frac{\log(\epsilon)}{\log(\gamma)}$. The coordinate-wise algorithm applied in this setting converges in exactly one iteration.*

Section 2.8.1, in the text describes empirical results and comparisons showing the performance of *SparseNet* as an optimization algorithm. In the spirit of Figure 2.7, attached below is a figure showing the objective value curves for all the 24 simulation examples.

Part II

Matrix Completion & Collaborative Filtering

Chapter 3

Spectral Regularization Algorithms

Abstract: In this chaper, we use convex relaxation techniques to provide a sequence of regularized low-rank solutions for large-scale matrix completion problems. Using the nuclear norm as a regularizer, we provide a simple and very efficient convex algorithm for minimizing the reconstruction error subject to a bound on the nuclear norm. Our algorithm SOFT-IMPUTE iteratively replaces the missing elements with those obtained from a soft-thresholded SVD. With warm starts this allows us to efficiently compute an entire regularization path of solutions on a grid of values of the regularization parameter. The computationally intensive part of our algorithm is in computing a low-rank SVD of a dense matrix. Exploiting the problem structure, we show that the task can be performed with a complexity of order linear in the matrix dimensions. Our semidefinite-programming algorithm is readily scalable to large matrices; for example SOFT-IMPUTE takes a few hours to compute low-rank approximations of a $10^6 \times 10^6$ incomplete matrix with 10^7 observed entries, and fits a rank-95 approximation to the full Netflix training set in 3.3 hours. Our methods achieve good training and test errors and exhibit superior timings when compared to other competitive state-of-the-art techniques.

This chapter is based on joint work with Trevor Hastie and Robert Tibshirani; and appears in our paper Mazumder et al. (2010).

3.1 Introduction

In many applications measured data can be represented in a matrix $X_{m\times n}$, for which only a relatively small number of entries are observed. The problem is to "complete" the matrix based on the observed entries, and has been dubbed the matrix completion problem (Candès & Recht 2008, Candès & Tao 2009, Rennie & Srebro 2005). The "Netflix" competition (SIGKDD & Netflix 2007, for example,) is a popular example, where the data is the basis for a recommender system. The rows correspond to viewers and the columns to movies, with the entry X_{ij} being the rating $\in \{1, \ldots, 5\}$ by viewer i for movie j. There are about 480K viewers and 18K movies, and hence 8.6 billion (8.6×10^9) potential entries. However, on average each viewer rates about 200 movies, so only 1.2% or 10^8 entries are observed. The task is to predict the ratings that viewers would give to movies they have not yet rated.

These problems can be phrased as learning an unknown parameter (a matrix $Z_{m\times n}$) with very high dimensionality, based on very few observations. In order for such inference to be meaningful, we assume that the parameter Z lies in a much lower dimensional manifold. In this chapter, as is relevant in many real life applications, we assume that Z can be well represented by a matrix of low rank, that is, $Z \approx V_{m\times k}G_{k\times n}$, where $k \ll \min(n, m)$. In this recommender-system example, low rank structure suggests that movies can be grouped into a small number of "genres", with $G_{\ell j}$ the relative score for movie j in genre ℓ. Viewer i on the other hand has an affinity $V_{i\ell}$ for genre ℓ, and hence the modeled score for viewer i on movie j is the sum $\sum_{\ell=1}^{k} V_{i\ell}G_{\ell j}$ of genre affinities times genre scores. Typically we view the observed entries in X as the corresponding entries from Z contaminated with noise.

Srebro, Alon & Jaakkola (2005) studied generalization error bounds for learning low-rank matrices. Recently Candès & Recht (2008), Candès & Tao (2009), and Keshavan et al. (2009) showed theoretically that under certain assumptions on the entries of the matrix, locations, and proportion of unobserved entries, the true underlying matrix can be recovered within very high accuracy.

For a matrix $X_{m\times n}$ let $\Omega \subset \{1,\ldots,m\}\times\{1,\ldots,n\}$ denote the indices of observed entries. We consider the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \text{rank}(Z) \\ \text{subject to} \quad & \sum_{(i,j)\in\Omega} (X_{ij} - Z_{ij})^2 \leq \delta, \end{aligned} \tag{3.1}$$

where $\delta \geq 0$ is a regularization parameter controlling the tolerance in training error. The rank constraint in (3.1) makes the problem for general Ω combinatorially hard (Srebro & Jaakkola 2003). For a fully-observed X on the other hand, the solution is given by a truncated singular value decomposition (SVD) of X. The following seemingly small modification to (3.1),

$$\begin{aligned} \text{minimize} \quad & \|Z\|_* \\ \text{subject to} \quad & \sum_{(i,j)\in\Omega} (X_{ij} - Z_{ij})^2 \leq \delta, \end{aligned} \tag{3.2}$$

makes the problem convex (Fazel 2002). Here $\|Z\|_*$ is the nuclear norm, or the sum of the singular values of Z. Under many situations the nuclear norm is an effective convex relaxation to the rank constraint (Fazel 2002, Candès & Recht 2008, Candès & Tao 2009, Recht et al. 2007). Optimization of (3.2) is a semi-definite programming problem (Boyd & Vandenberghe 2004) and can be solved efficiently for small problems, using modern convex optimization software like SeDuMi and SDPT3 (Grant & Boyd. 2009). However, since these algorithms are based on second order methods (Liu & Vandenberghe 2009), they can become prohibitively expensive if the dimensions of the matrix get large (Cai et al. 2010). Equivalently we can reformulate (3.2) in *Lagrange* form

$$\underset{Z}{\text{minimize}}\ \frac{1}{2} \sum_{(i,j)\in\Omega} (X_{ij} - Z_{ij})^2 + \lambda \|Z\|_*. \tag{3.3}$$

Here $\lambda \geq 0$ is a regularization parameter controlling the nuclear norm of the minimizer $\hat{Z}_\lambda$ of (3.3); there is a 1-1 mapping between $\delta \geq 0$ and $\lambda \geq 0$ over their active domains.

In this chapter we propose an algorithm Soft-Impute for the nuclear norm regularized least-squares problem (3.3) that scales to large problems with $m, n \approx 10^5$–10^6 with around 10^6–10^8 or more observed entries. At every iteration Soft-Impute decreases the value of the objective function towards its minimum, and at the same time gets closer to the set of optimal solutions of the problem (3.2). We study the convergence properties of this algorithm and discuss how it can be extended to other more sophisticated forms of spectral regularization.

To summarize some performance results[1]

- We obtain a rank-40 solution to (3.2) for a problem of size $10^5 \times 10^5$ and $|\Omega| = 5 \times 10^6$ observed entries in less than 18 minutes.
- For the same sized matrix with $|\Omega| = 10^7$ we obtain a rank-5 solution in less than 21 minutes.
- For a $10^6 \times 10^5$ sized matrix with $|\Omega| = 10^8$ a rank-5 solution is obtained in approximately 4.3 hours.
- We fit a rank-66 solution for the Netflix data in 2.2 hours. Here there are 10^8 observed entries in a matrix with 4.8×10^5 rows and 1.8×10^4 columns. A rank 95 solution takes 3.27 hours.

The chapter is organized as follows. In Section 3.2, we discuss related work and provide some context for this chapter. In Section 3.3 we introduce the Soft-Impute algorithm and study its convergence properties in Section 3.4. The computational aspects of the algorithm are described in Section 3.5, and Section 3.6 discusses how nuclear norm regularization can

[1] For large problems data transfer, access and reading take quite a lot of time and is dependent upon the platform and machine. Over here we report the times taken for the computational bottle-neck, that is, the SVD computations over all iterations. All times are reported based on computations done in a Intel Xeon Linux 3GHz processor using MATLAB, with no C or Fortran interlacing.

be generalized to more aggressive and general types of spectral regularization. Section 3.7 describes post-processing of "selectors" and initialization. We discuss comparisons with related work, simulations and experimental studies in Section 3.9 and application to the Netflix data in Section 3.10.

3.2 Context and Related Work

Candès & Tao (2009), Cai et al. (2010), and Candès & Recht (2008) consider the criterion

$$\begin{aligned} \text{minimize} \quad & \|Z\|_* \\ \text{subject to} \quad & Z_{ij} = X_{ij},\ \forall (i,j) \in \Omega. \end{aligned} \tag{3.4}$$

With $\delta = 0$, the criterion (3.1) is equivalent to (3.4), in that it requires the training error to be zero. Cai et al. (2010) propose a first-order singular-value-thresholding algorithm SVT scalable to large matrices for the problem (3.4). They comment on the problem (3.2) with $\delta > 0$, but dismiss it as being computationally prohibitive for large problems.

We believe that (3.4) will almost always be too rigid and will result in over-fitting. If minimization of prediction error is an important goal, then the optimal solution $\hat{Z}$ will typically lie somewhere in the interior of the path indexed by δ (Figures 3.2, 3.3 and 3.4).

In this chapter we provide an algorithm Soft-Impute for computing solutions of (3.3) on a grid of λ values, based on warm restarts. The algorithm is inspired by SVD-Impute (Troyanskaya et al. 2001)—an EM-type (Dempster et al. 1977) iterative algorithm that alternates between imputing the missing values from a current SVD, and updating the SVD using the "complete" data matrix. In its very motivation, Soft-Impute is different from generic first order algorithms (Cai et al. 2010, Ma et al. 2011, Ji & Ye 2009). The latter require the specification of a step size, and can be quite sensitive to the chosen value. Our algorithm does not require a step-size, or any such parameter.

The iterative algorithms proposed in Ma et al. (2011) and Ji & Ye (2009) require the computation of a SVD of a dense matrix (with dimensions equal to the size of the matrix X) at every iteration, as the bottleneck. This makes the algorithms prohibitive for large scale computations. Ma et al. (2011) use randomized algorithms for the SVD computation. Our algorithm SOFT-IMPUTE also requires an SVD computation at every iteration, but by exploiting the *problem structure*, can easily handle matrices of very large dimensions. At each iteration the non-sparse matrix has the structure:

$$Y = Y_{SP} \;\text{(Sparse)} \quad + \quad Y_{LR} \;\text{(Low Rank)}. \tag{3.5}$$

In (3.5) Y_{SP} has the same sparsity structure as the observed X, and Y_{LR} has rank $\tilde{r} \ll m, n$, where $\tilde{r}$ is very close to $r \ll m, n$ the rank of the estimated matrix Z (upon convergence of the algorithm). For large scale problems, we use iterative methods based on Lanczos bidiagonalization with partial re-orthogonalization (as in the PROPACK algorithm, (Larsen 1998)), for computing the first $\tilde{r}$ singular vectors/values of Y. Due to the specific structure of (3.5), multiplication by Y and Y' can both be achieved in a cost-efficient way. In decomposition (3.5), the computationally burdensome work in computing a low-rank SVD is of an order that depends linearly on the matrix dimensions. More precisely, evaluating each singular vector requires computation of the order of $O((m+n)\tilde{r}) + O(|\Omega|)$ flops and evaluating r' of them requires $O((m+n)\tilde{r}r') + O(|\Omega|r')$ flops. Exploiting warm-starts, we observe that $\tilde{r} \approx r$—hence every SVD step of our algorithm computes r singular vectors, with complexity of the order $O((m+n)r^2) + O(|\Omega|r)$ flops. This computation is performed for the number of iterations SOFT-IMPUTE requires to run till convergence or a certain tolerance.

In this chapter we show asymptotic convergence of SOFT-IMPUTE and further derive its non-asymptotic rate of convergence which scales as $O(1/k)$ (k denotes the iteration number). However, in our experimental studies on low-rank matrix completion, we have observed that our algorithm is faster (based on timing comparisons) than the accelerated version of

Nesterov (Ji & Ye 2009, Nesterov 2007), having a provable (worst case) convergence rate of $O(\frac{1}{k^2})$. With warm-starts SOFT-IMPUTE computes the entire regularization path very efficiently along a dense series of values for λ.

Although the nuclear norm is motivated here as a convex relaxation to a rank constraint, we believe in many situations it will outperform the rank-restricted estimator (3.1). This is supported by our experimental studies. We draw the natural analogy with model selection in linear regression, and compare best-subset regression (ℓ_0 regularization) with the LASSO (Tibshirani 1996, Hastie et al. 2009, ℓ_1 regularization,). There too the ℓ_1 penalty can be viewed as a convex relaxation of the ℓ_0 penalty. But in many situations with moderate sparsity, the LASSO will outperform best subset in terms of prediction accuracy (Friedman 2008, Hastie et al. 2009, Mazumder et al. 2011). By shrinking the parameters in the model (and hence reducing their variance), the lasso permits more parameters to be included. The nuclear norm is the ℓ_1 penalty in matrix completion, as compared to the ℓ_0 rank. By shrinking the singular values, we allow more dimensions to be included without incurring undue estimation variance.

Another class of techniques used in collaborative filtering problems are close in spirit to (3.2). These are known as *maximum margin matrix factorization* methods—in short MMMF—and use a factor model for the matrix Z (Srebro, Rennie & Jaakkola 2005). Let $Z = UV'$ where $U_{m\times r'}$ and $V_{n\times r'}$, and consider the following problem

$$\underset{U,V}{\text{minimize}}\, \frac{1}{2} \sum_{(i,j)\in\Omega} (X_{ij} - (UV')_{ij})^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2). \tag{3.6}$$

It turns out that (3.6) is intimately related to (3.3), since (see Lemma 11)

$$||Z||_* = \underset{U,V:\, Z=UV'}{\text{minimize}}\, \frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right).$$

For example, if $r' = \min(m, n)$, the solution to (3.6) coincides with the solution to (3.3).[2] However, (3.6) is not convex in its arguments, while (3.3) is. We compare these two criteria in detail in Section 3.8, and the relative performance of their respective algorithms in Section 3.9.2.

3.3 Soft-Impute–an Algorithm for Nuclear Norm Regularization

We first introduce some notation that will be used for the rest of this article.

3.3.1 Notation

We adopt the notation of Cai et al. (2010). Define a matrix $P_\Omega(Y)$ (with dimension $m \times n$)

$$P_\Omega(Y)\,(i,j) = \begin{cases} Y_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{if } (i,j) \notin \Omega, \end{cases} \tag{3.7}$$

which is a projection of the matrix $Y_{m\times n}$ onto the observed entries. In the same spirit, define the complementary projection $P_\Omega^\perp(Y)$ via $P_\Omega^\perp(Y) + P_\Omega(Y) = Y$. Using (3.7) we can rewrite $\sum_{(i,j)\in\Omega}(X_{ij} - Z_{ij})^2$ as $\|P_\Omega(X) - P_\Omega(Z)\|_F^2$.

3.3.2 Nuclear Norm Regularization

We present the following lemma, which forms a basic ingredient in our algorithm.

Lemma 6. *Suppose the matrix $W_{m\times n}$ has rank r. The solution to the optimization problem*

$$\underset{Z}{\text{minimize}} \quad \tfrac{1}{2}\|W - Z\|_F^2 + \lambda\|Z\|_* \tag{3.8}$$

[2] We note here that the original MMMF formulation uses $r' = \min\{m, n\}$. In this chapter we will consider it for a family of r' values.

is given by $\hat{Z} = \mathrm{S}_\lambda(W)$ *where*

$$\mathrm{S}_\lambda(W) \equiv UD_\lambda V' \quad \textit{with} \quad D_\lambda = \mathrm{diag}\left[(d_1 - \lambda)_+, \ldots, (d_r - \lambda)_+\right], \tag{3.9}$$

UDV' *is the SVD of* W, $D = \mathrm{diag}\left[d_1, \ldots, d_r\right]$, *and* $t_+ = \max(t, 0)$.

The notation $\mathrm{S}_\lambda(W)$ refers to *soft-thresholding* (Donoho et al. 1995). Lemma 6 appears in Cai et al. (2010) and Ma et al. (2011) where the proof uses the sub-gradient characterization of the nuclear norm. In Appendix 3.A.1 we present an entirely different proof, which can be extended in a relatively straightforward way to other complicated forms of spectral regularization discussed in Section 3.6. Our proof is followed by a remark that covers these more general cases.

3.3.3 Algorithm

Using the notation in 3.3.1, we rewrite (3.3) as:

$$\underset{Z}{\mathrm{minimize}} \quad f_\lambda(Z) := \tfrac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda\|Z\|_* \, . \tag{3.10}$$

We now present Algorithm 2—Soft-Impute—for computing a series of solutions to (3.10) for different values of λ using warm starts.

The algorithm repeatedly replaces the missing entries with the current guess, and then updates the guess by solving (3.8). Figures 3.2, 3.3 and 3.4 show some examples of solutions using Soft-Impute (blue continuous curves). We see test and training error in the top rows as a function of the nuclear norm, obtained from a grid of values Λ. These error curves show a smooth and very competitive performance.

Algorithm 2 SOFT-IMPUTE

1. Initialize $Z^{\text{old}} = 0$.
2. Do for $\lambda_1 > \lambda_2 > \ldots > \lambda_K$:
   (a) Repeat:
      i. Compute $Z^{\text{new}} \leftarrow \mathrm{S}_{\lambda_k}(P_\Omega(X) + P_\Omega^\perp(Z^{\text{old}}))$.
      ii. If $\frac{\|Z^{\text{new}} - Z^{\text{old}}\|_F^2}{\|Z^{\text{old}}\|_F^2} < \epsilon$ exit.
      iii. Assign $Z^{\text{old}} \leftarrow Z^{\text{new}}$.
   (b) Assign $\hat{Z}_{\lambda_k} \leftarrow Z^{\text{new}}$.
3. Output the sequence of solutions $\hat{Z}_{\lambda_1}, \ldots, \hat{Z}_{\lambda_K}$.

3.4 Convergence Analysis

In this section we study the convergence properties of Algorithm 2. Unlike generic first-order methods (Nesterov 2003) including competitive first-order methods for nuclear norm regularized problems (Cai et al. 2010, Ma et al. 2011), SOFT-IMPUTE does not involve the choice of any additional step-size. Most importantly our algorithm is readily scalable for solving large scale semidefinite programming problems (3.2) and (3.10) as will be explained later in Section 3.5.

For an arbitrary matrix $\tilde{Z}$, define

$$Q_\lambda(Z|\tilde{Z}) = \tfrac{1}{2}\|P_\Omega(X) + P_\Omega^\perp(\tilde{Z}) - Z\|_F^2 + \lambda\|Z\|_* \tag{3.11}$$

as a surrogate of the objective function $f_\lambda(z)$. Note that $f_\lambda(\tilde{Z}) = Q_\lambda(\tilde{Z}|\tilde{Z})$ for any $\tilde{Z}$.

In Section 3.4.1, we show that the sequence Z_λ^k generated via SOFT-IMPUTE *converges* asymptotically, that is, as $k \to \infty$ to a minimizer of the objective function $f_\lambda(Z)$. SOFT-IMPUTE produces a sequence of solutions for which the criterion decreases to the optimal solution with every iteration and the successive iterates get closer to the optimal set of solutions of the problem 3.10. Section 3.4.2 derives the non-asymptotic convergence rate of the algorithm. The latter analysis concentrates on the objective values $f_\lambda(Z^k_\lambda)$. Due to computational resources if one wishes to stop the algorithm after K iterations, then Theorem 7 provides a certificate of how *far* Z^k_λ is from the solution. Though Section 3.4.1 alone establishes the convergence of $f_\lambda(Z^k_\lambda)$ to the minimum of $f_\lambda(Z)$, this does not, in general, settle the *convergence* of Z^k_λ unless further conditions (like strong convexity) are imposed on $f_\lambda(\cdot)$.

3.4.1 Asymptotic Convergence

Lemma 7. *For every fixed $\lambda \geq 0$, define a sequence Z^k_λ by*

$$Z^{k+1}_\lambda = \arg\min_Z Q_\lambda(Z|Z^k_\lambda)$$

with any starting point Z^0_λ. The sequence Z^k_λ satisfies

$$f_\lambda(Z^{k+1}_\lambda) \leq Q_\lambda(Z^{k+1}_\lambda|Z^k_\lambda) \leq f_\lambda(Z^k_\lambda).$$

Proof. Note that

$$Z^{k+1}_\lambda = \mathrm{S}_\lambda(P_\Omega(X) + P^\perp_\Omega(Z^k_\lambda)). \tag{3.12}$$

By Lemma 6 and the definition (3.11) of $Q_\lambda(Z|Z_\lambda^k)$, we have:

$$\begin{aligned}
f_\lambda(Z_\lambda^k) &= Q_\lambda(Z_\lambda^k|Z_\lambda^k) \\
&= \tfrac{1}{2}\|P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k) - Z_\lambda^k\|_F^2 + \lambda\|Z_\lambda^k\|_* \\
&\geq \min_Z \tfrac{1}{2}\left\{\|P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k) - Z\|_F^2\right\} + \lambda\|Z\|_* \\
&= Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^k) \\
&= \tfrac{1}{2}\|\left\{P_\Omega(X) - P_\Omega(Z_\lambda^{k+1})\right\} + \left\{P_\Omega^\perp(Z_\lambda^k) - P_\Omega^\perp(Z_\lambda^{k+1})\right\}\|_F^2 + \lambda\|Z_\lambda^{k+1}\|_* \\
&= \tfrac{1}{2}\left\{\|P_\Omega(X) - P_\Omega(Z_\lambda^{k+1})\|_F^2 + \|P_\Omega^\perp(Z_\lambda^k) - P_\Omega^\perp(Z_\lambda^{k+1})\|_F^2\right\} \\
&\quad + \lambda\|Z_\lambda^{k+1}\|_* \qquad (3.13) \\
&\geq \tfrac{1}{2}\|P_\Omega(X) - P_\Omega(Z_\lambda^{k+1})\|_F^2 + \lambda\|Z_\lambda^{k+1}\|_* \qquad (3.14) \\
&= Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^{k+1}) \\
&= f(Z_\lambda^{k+1}).
\end{aligned}$$

□

Lemma 8. *The nuclear norm shrinkage operator* $\mathrm{S}_\lambda(\cdot)$ *satisfies the following for any* W_1, W_2 *(with matching dimensions)*

$$\|\mathrm{S}_\lambda(W_1) - \mathrm{S}_\lambda(W_2)\|_F^2 \leq \|W_1 - W_2\|_F^2.$$

In particular this implies that $\mathrm{S}_\lambda(W)$ *is a continuous map in* W*.*

Lemma 8 is proved in Ma et al. (2011); their proof is complex and based on trace inequalities. We give a concise proof based on elementary convex analysis in Appendix 3.A.2.

Lemma 9. *The successive differences* $\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2$ *of the sequence* Z_λ^k *are monotone decreasing:*

$$\|Z_\lambda^{k+1} - Z_\lambda^k\|_F^2 \leq \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2 \quad \forall k. \qquad (3.15)$$

Moreover the difference sequence converges to zero. That is

$$Z_\lambda^{k+1} - Z_\lambda^k \to 0 \text{ as } k \to \infty.$$

The proof of Lemma 9 is given in Appendix 3.A.3.

Lemma 10. *Every limit point of the sequence Z_λ^k defined in Lemma 7 is a stationary point of*

$$\tfrac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda\|Z\|_*.$$

Hence it is a solution to the fixed point equation

$$Z = \mathrm{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z)). \tag{3.16}$$

The proof of Lemma 10 is given in Appendix 3.A.4.

Theorem 6. *The sequence Z_λ^k defined in Lemma 7 converges to a limit Z_λ^∞ that solves*

$$\underset{Z}{\text{minimize}} \quad \tfrac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda\|Z\|_*. \tag{3.17}$$

Proof. It suffices to prove that Z_λ^k converges; the theorem then follows from Lemma 10.

Let $\hat{Z}_\lambda$ be a limit point of the sequence Z_λ^k. There exists a subsequence m_k such that $Z_\lambda^{m_k} \to \hat{Z}_\lambda$. By Lemma 10, $\hat{Z}_\lambda$ solves the problem (3.17) and satisfies the fixed point equation (3.16).

Hence

$$\begin{aligned} \|\hat{Z}_\lambda - Z_\lambda^k\|_F^2 &= \|\mathrm{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(\hat{Z}_\lambda)) - \mathrm{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))\|_F^2 && (3.18) \\ &\leq \|(P_\Omega(X) + P_\Omega^\perp(\hat{Z}_\lambda)) - (P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))\|_F^2 \\ &= \|P_\Omega^\perp(\hat{Z}_\lambda - Z_\lambda^{k-1})\|_F^2 \\ &\leq \|\hat{Z}_\lambda - Z_\lambda^{k-1}\|_F^2. && (3.19) \end{aligned}$$

In (3.18) two substitutions were made; the left one using (3.16) in Lemma 10, the right one using (3.12). Inequality (3.19) implies that the sequence $\|\hat{Z}_\lambda - Z_\lambda^{k-1}\|_F^2$ converges as $k \to \infty$. To show the convergence of the sequence Z_λ^k it suffices to prove that the sequence $\hat{Z}_\lambda - Z_\lambda^k$ converges to zero. We prove this by contradiction.

Suppose the sequence Z_λ^k has another limit point $Z_\lambda^+ \neq \hat{Z}_\lambda$. Then $\hat{Z}_\lambda - Z_\lambda^k$ has two distinct limit points 0 and $Z_\lambda^+ - \hat{Z}_\lambda \neq 0$. This contradicts the convergence of the sequence $\|\hat{Z}_\lambda - Z_\lambda^{k-1}\|_F^2$. Hence the sequence Z_λ^k converges to $\hat{Z}_\lambda := Z_\lambda^\infty$. □

The inequality in (3.19) implies that at every iteration Z_λ^k gets closer to an optimal solution for the problem (3.17).[3] This property holds in addition to the decrease of the objective function (Lemma 7) at every iteration.

3.4.2 Convergence Rate

In this section we derive the worst case convergence rate of SOFT-IMPUTE.

Theorem 7. *For every fixed $\lambda \geq 0$, the sequence $Z_\lambda^k; k \geq 0$ defined in Lemma 7 has the following non-asymptotic (worst) rate of convergence:*

$$f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty) \leq \frac{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2}{k+1}. \tag{3.20}$$

The proof of this theorem is in Appendix 3.A.6.

In light of Theorem 7, a $\delta > 0$ accurate solution of $f_\lambda(Z)$ is obtained after a maximum of $\frac{2}{\delta}\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2$ iterations. Using warm-starts, SOFT-IMPUTE traces out the path of solutions on a grid of λ values $\lambda_1 > \lambda_2 > \ldots > \lambda_K$ with a total of [4]

$$\sum_{i=1}^{K} \frac{2}{\delta}\|\hat{Z}_{\lambda_{i-1}} - Z_{\lambda_i}^\infty\|_F^2 \tag{3.21}$$

[3] In fact this statement can be strengthened further—at every iteration the distance of the estimate decreases from the set of optimal solutions.

[4] We assume the solution $\hat{Z}_\lambda$ at every $\lambda \in \{\lambda_1, \ldots, \lambda_K\}$ is computed to an accuracy of $\delta > 0$.

iterations. Here $\hat{Z}_{\lambda_0} = 0$ and $\hat{Z}_{\lambda_i}$ denotes the output of SOFT-IMPUTE (upon convergence) for $\lambda = \lambda_i$ ($i \in \{1, \ldots, K-1\}$). The solutions $Z^{\infty}_{\lambda_i}$ and $Z^{\infty}_{\lambda_{i-1}}$ are likely to be *close* to each other, especially on a dense grid of λ_i's. Hence every summand of (3.21) and the total number of iterations is expected to be significantly smaller than that obtained via arbitrary cold-starts.

3.5 Computational Complexity

The computationally demanding part of Algorithm 2 is in $\mathrm{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k))$. This requires calculating a low-rank SVD of a matrix, since the underlying model assumption is that $\mathrm{rank}(Z) \ll \min\{m, n\}$. In Algorithm 2, for fixed λ, the entire sequence of matrices Z_λ^k have explicit[5] low-rank representations of the form $U_k D_k V_k'$ corresponding to $\mathrm{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))$.

In addition, observe that $P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k)$ can be rewritten as

$$\begin{aligned} P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k) &= \left\{P_\Omega(X) - P_\Omega(Z_\lambda^k)\right\} + Z_\lambda^k \\ &= \textit{Sparse} + \textit{Low Rank}. \end{aligned} \tag{3.22}$$

In the numerical linear algebra literature, there are very efficient direct matrix factorization methods for calculating the SVD of matrices of moderate size (at most a few thousand). When the matrix is sparse, larger problems can be solved but the computational cost depends heavily upon the sparsity structure of the matrix. In general however, for large matrices one has to resort to indirect iterative methods for calculating the leading singular vectors/values of a matrix. There is a lot research in numerical linear algebra for developing sophisticated algorithms for this purpose. In this chapter we will use the PROPACK algorithm (Larsen 2004, Larsen 1998) because of its low storage requirements, effective flop

[5] Though we cannot prove theoretically that every iterate of the sequence Z_λ^k will be of low-rank; this observation is rather practical based on the manner in which we trace out the entire path of solutions based on warm-starts. Our simulation results support this observation as well.

count and its well documented MATLAB version. The algorithm for calculating the truncated SVD for a matrix W (say), becomes efficient if multiplication operations Wb_1 and $W'b_2$ (with $b_1 \in \Re^n,\ b_2 \in \Re^m$) can be done with minimal cost.

Algorithm SOFT-IMPUTE requires repeated computation of a truncated SVD for a matrix W with structure as in (3.22). Assume that at the current iterate, the matrix Z_λ^k has rank $\tilde{r}$. Note that in (3.22) the term $P_\Omega(Z_\lambda^k)$ can be computed in $O(|\Omega|\tilde{r})$ flops using only the required outer products (i.e., our algorithm does not compute the matrix explicitly).

The cost of computing the truncated SVD will depend upon the cost in the operations Wb_1 and $W'b_2$ (which are equal). For the sparse part these multiplications cost $O(|\Omega|)$. Although it costs $O(|\Omega|\tilde{r})$ to create the matrix $P_\Omega(Z_\lambda^k)$, this is used for each of the $\tilde{r}$ such multiplications (which also cost $O(|\Omega|\tilde{r})$), so we need not include that cost here. The Low Rank part costs $O((m+n)\tilde{r})$ for the multiplication by b_1. Hence the cost is $O(|\Omega|) + O((m+n)\tilde{r})$ per vector multiplication. Supposing we want a $\tilde{r}$ rank SVD of the matrix (3.22), the cost will be of the order of $O(|\Omega|\tilde{r}) + O((m+n)(\tilde{r})^2)$ (for that iteration, that is, to obtain Z_λ^{k+1} from Z_λ^k). Suppose the rank of the solution Z_λ^k is r, then in light of our above observations $\tilde{r} \approx r \ll \min\{m,n\}$ and the order is $O(|\Omega|r) + O((m+n)r^2)$.

For the reconstruction problem to be theoretically meaningful in the sense of Candès & Tao (2009) we require that $|\Omega| \approx nr \cdot \mathrm{poly}(\log n)$. In practice often $|\Omega|$ is very small. Hence introducing the *Low Rank* part does not add any further complexity in the multiplication by W and W'. So the dominant cost in calculating the truncated SVD in our algorithm is $O(|\Omega|)$. The SVT algorithm (Cai et al. 2010) for exact matrix completion (3.4) involves calculating the SVD of a sparse matrix with cost $O(|\Omega|)$. This implies that the computational order of SOFT-IMPUTE and that of SVT is the same. This order computation does not include the number of iterations required for convergence. In our experimental studies we use warm-starts for efficiently computing the entire regularization path. On small scale examples, based on comparisons with the accelerated gradient method of Nesterov (Ji & Ye 2009, Nesterov 2007, see Section 3.9.3;) we find that our algorithm converges faster than the latter in terms of run-time and number of SVD computations/ iterations. This supports the computational effectiveness of SOFT-IMPUTE. In addition, since the true rank of the matrix $r \ll \min\{m, n\}$, the computational cost of evaluating the truncated SVD (with rank $\approx r$) is linear in matrix dimensions. This justifies the large-scale computational feasibility of our algorithm.

The above discussions focus on the computational complexity for obtaining a low-rank SVD, which is to be performed at every iteration of SOFT-IMPUTE. Similar to the total iteration complexity bound of SOFT-IMPUTE (3.21), the total cost to compute the regularization path on a grid of λ values is given by:

$$\sum_{i=1}^{K} O\left((|\Omega|\bar{r}_{\lambda_i} + (m+n)\bar{r}^2_{\lambda_i}) \frac{2}{\delta} \|\hat{Z}_{\lambda_{i-1}} - Z^{\infty}_{\lambda_i}\|_F^2 \right).$$

Here $\bar{r}_\lambda$ denotes the rank[6] (on an average) of the iterates Z^k_λ generated by SOFT-IMPUTE for fixed λ.

The PROPACK package does not allow one to request (and hence compute) only the singular values larger than a threshold λ—one has to specify the number in advance. So once all the computed singular values fall above the current threshold λ, our algorithm increases the number to be computed until the smallest is smaller than λ. In large scale problems, we put an absolute limit on the maximum number.

3.6 Generalized Spectral Regularization: From Soft to Hard Thresholding

In Section 3.1 we discussed the role of the nuclear norm as a convex surrogate for the rank of a matrix, and drew the analogy with LASSO regression versus best-subset selection. We argued that in many problems ℓ_1 regularization gives better prediction accuracy. However,

[6]We assume, above that the grid of values $\lambda_1 > \ldots \lambda_K$ is such that *all* the solutions $Z_\lambda, \lambda \in \{\lambda_1, \ldots, \lambda_K\}$ are of *small* rank, as they appear in Section 3.5.

if the underlying model is very sparse, then the LASSO with its uniform shrinkage can both overestimate the number of non-zero coefficients (Friedman 2008) in the model, and overly shrink (bias) those included toward zero. In this section we propose a natural generalization of SOFT-IMPUTE to overcome these problems.

Consider again the problem

$$\underset{\text{rank}(Z)\leq k}{\text{minimize}} \quad \tfrac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2,$$

a rephrasing of (3.1). This best rank-k solution also solves

$$\text{minimize}\, \tfrac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda \sum_j I(\gamma_j(Z) > 0),$$

where $\gamma_j(Z)$ is the jth singular value of Z, and for a suitable choice of λ that produces a solution with rank k.

The "fully observed" matrix version of the above problem is given by the ℓ_0 version of (3.8) as follows:

$$\underset{Z}{\text{minimize}} \quad \tfrac{1}{2}\|W - Z\|_F^2 + \lambda\|Z\|_0, \tag{3.23}$$

where $\|Z\|_0 = \text{rank}(Z)$. The solution of (3.23) is given by a reduced-rank SVD of W; for every λ there is a corresponding $q = q(\lambda)$ number of singular-values to be retained in the SVD decomposition. Problem (3.23) is non-convex in W but its global minimizer can be evaluated. As in (3.9) the thresholding operator resulting from (3.23) is

$$\mathrm{S}_\lambda^H(W) = UD_qV' \quad \text{where} \quad D_q = \text{diag}\left(d_1, \ldots, d_q, 0, \ldots, 0\right).$$

Similar to SOFT-IMPUTE (Algorithm 2), we present below HARD-IMPUTE (Algorithm 3) for the ℓ_0 penalty. The continuous parameterization via λ does not appear to offer obvious

Algorithm 3 HARD-IMPUTE

1. Initialize $\tilde{Z}_{\lambda_k}$ $k = 1, \ldots, K$ (for example, using SOFT-IMPUTE; see Section 3.7).
2. Do for $\lambda_1 > \lambda_2 > \ldots > \lambda_K$:
   (a) Repeat:
      i. Compute $Z^{\text{new}} \leftarrow \mathrm{S}^H_{\lambda_k}(P_\Omega(X) + P^\perp_\Omega(Z^{\text{old}}))$.
      ii. If $\frac{\|Z^{\text{new}} - Z^{\text{old}}\|_F^2}{\|Z^{\text{old}}\|_F^2} < \epsilon$ exit.
      iii. Assign $Z^{\text{old}} \leftarrow Z^{\text{new}}$.
   (b) Assign $\hat{Z}_{H,\lambda_k} \leftarrow Z^{\text{new}}$.
3. Output the sequence of solutions $\hat{Z}_{H,\lambda_1}, \ldots, \hat{Z}_{H,\lambda_K}$.

advantages over rank-truncation methods. We note that it does allow for a continuum of warm starts, and is a natural post-processor for the output of SOFT-IMPUTE (next section). But it also allows for further generalizations that bridge the gap between hard and soft regularization methods.

In penalized regression there have been recent developments directed towards "bridging" the gap between the ℓ_1 and ℓ_0 penalties (Friedman 2008, Zhang 2010, Mazumder et al. 2011). This is done via using non-convex penalties that are a better surrogate (in the sense of approximating the penalty) to ℓ_0 over the ℓ_1. They also produce less biased estimates than those produced by the ℓ_1 penalized solutions. When the underlying model is very sparse they often perform very well, and enjoy superior prediction accuracy when compared to softer penalties like ℓ_1. These methods still shrink, but are less aggressive than the best-subset selection.

By analogy, we propose using a more sophisticated version of spectral regularization. This goes beyond nuclear norm regularization by using slightly more aggressive penalties that bridge the gap between ℓ_1 (nuclear norm) and ℓ_0(rank constraint). We propose minimizing

$$f_{\mathbf{p},\lambda}(Z) \quad = \quad \tfrac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda \sum_j \mathbf{p}(\gamma_j(Z);\mu), \tag{3.24}$$

where $\mathbf{p}(|t|;\mu)$ is concave in $|t|$. The parameter $\mu \in [\mu_{\text{inf}}, \mu_{\text{sup}}]$ controls the degree of concavity. We may think of $p(|t|;\mu_{\text{inf}}) = |t|$ (ℓ_1 penalty) on one end and $p(|t|;\mu_{\text{sup}}) = \|t\|_0$ (ℓ_0 penalty) on the other. In particular for the ℓ_0 penalty denote $f_{\mathbf{p},\lambda}(Z)$ by $f_{H,\lambda}(Z)$ for "hard" thresholding. See Friedman (2008), Mazumder et al. (2011) and Zhang (2010) for examples of such penalties.

In Remark 1 in Appendix 3.A.1 we argue how the proof can be modified for general types of spectral regularization. Hence for minimizing the objective (3.24) we will look at the analogous version of (3.8, 3.23) which is

$$\underset{Z}{\text{minimize}} \quad \tfrac{1}{2}\|W - Z\|_F^2 + \lambda \sum_j \mathbf{p}(\gamma_j(Z);\mu).$$

The solution is given by a thresholded SVD of W,

$$\mathrm{S}_\lambda^{\mathbf{p}}(W) = UD_{\mathbf{p},\lambda}V',$$

where $D_{\mathbf{p},\lambda}$ is a entry-wise thresholding of the diagonal entries of the matrix D consisting of singular values of the matrix W. The exact form of the thresholding depends upon the form of the penalty function $\mathbf{p}(\cdot;\cdot)$, as discussed in Remark 1. Algorithm 2 and Algorithm 3 can be modified for the penalty $\mathbf{p}(\cdot;\mu)$ by using a more general thresholding function $\mathrm{S}_\lambda^{\mathrm{p}}(\cdot)$ in Step 2(a)i. The corresponding step becomes:

$$Z^{\text{new}} \leftarrow \mathrm{S}_\lambda^{\mathbf{p}}(P_\Omega(X) + P_\Omega^\perp(Z^{\text{old}})).$$

However these types of spectral regularization make the criterion (3.24) non-convex and hence it becomes difficult to optimize globally. Recht et al. (2007) and Bach (2008) also consider the rank estimation problem from a theoretical standpoint.

3.7 Post-processing of "Selectors" and Initialization

Because the ℓ_1 norm regularizes by shrinking the singular values, the number of singular values retained (through cross-validation, say) may exceed the actual rank of the matrix. In such cases it is reasonable to *undo* the shrinkage of the chosen models, which might permit a lower-rank solution.

If Z_λ is the solution to (3.10), then its *post-processed* version Z_λ^u obtained by "unshrinking" the eigen-values of the matrix Z_λ is obtained by

$$\begin{aligned} \boldsymbol{\alpha} &= \underset{\alpha_i \geq 0,\, i=1,\ldots,r_\lambda}{\arg\min} \quad \|P_\Omega(X) - \sum_{i=1}^{r_\lambda} \alpha_i P_\Omega(u_i v_i')\|^2 \\ Z_\lambda^u &= U D_\alpha V', \end{aligned} \tag{3.25}$$

where $D_\alpha = \mathrm{diag}(\alpha_1, \ldots, \alpha_{r_\lambda})$. Here r_λ is the rank of Z_λ and $Z_\lambda = UD_\lambda V'$ is its SVD. The estimation in (3.25) can be done via ordinary least squares, which is feasible because of the sparsity of $P_\Omega(u_i v_i')$ and that r_λ is small.[7] If the least squares solutions $\boldsymbol{\alpha}$ do not meet the positivity constraints, then the negative sign can be absorbed into the corresponding singular vector.

Rather than estimating a diagonal matrix D_α as above, one can insert a matrix $M_{r_\lambda \times r_\lambda}$ between U and V above to obtain better training error for the same rank. Hence given

[7] Observe that the $P_\Omega(u_i v_i')$, $i = 1, \ldots, r_\lambda$ are not orthogonal, though the $u_i v_i'$ are.

U, V (each of rank r_λ) from the SOFT-IMPUTE algorithm, we solve

$$\begin{aligned} \hat{M} &= \arg\min_{M} \quad \|P_\Omega(X) - P_\Omega(UMV')\|^2, \\ \text{where,} \quad \hat{Z}_\lambda &= U\hat{M}V'. \end{aligned} \tag{3.26}$$

The objective function in (3.26) is the Frobenius norm of an affine function of M and hence can be optimized very efficiently. Scalability issues pertaining to the optimization problem (3.26) can be handled fairly efficiently via conjugate gradients. Criterion (3.26) will definitely lead to a decrease in training error as that attained by $\hat{Z} = UD_\lambda V'$ for the same rank and is potentially an attractive proposal for the original problem (3.1). However this heuristic cannot be caste as a (jointly) convex problem in (U, M, V). In addition, this requires the estimation of up to r_λ^2 parameters, and has the potential for over-fitting. In this chapter we report experiments based on (3.25).

In many simulated examples we have observed that this post-processing step gives a good estimate of the underlying true rank of the matrix (based on prediction error). Since fixed points of Algorithm 3 correspond to local minima of the function (3.24), well-chosen warm starts $\tilde{Z}_\lambda$ are helpful. A reasonable prescription for warms-starts is the nuclear norm solution via (SOFT-IMPUTE), or the post processed version (3.25). The latter appears to significantly speed up convergence for HARD-IMPUTE. This observation is based on our simulation studies.

3.8 Soft-Impute and Maximum-Margin Matrix Factorization

In this section we compare in detail the MMMF criterion (3.6) with the SOFT-IMPUTE criterion (3.3). For ease of comparison here, we put down these criteria again using our P_Ω notation.

MMMF solves

$$\underset{U,V}{\text{minimize}} \quad \tfrac{1}{2}||P_{\Omega}(X - UV)||_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2), \tag{3.27}$$

where $U_{m \times r'}$ and $V_{n \times r'}$ are arbitrary (non-orthogonal) matrices. This problem formulation and related optimization methods have been explored by Srebro, Rennie & Jaakkola (2005) and Rennie & Srebro (2005).

Soft-Impute solves

$$\underset{Z}{\text{minimize}} \quad \tfrac{1}{2}||P_{\Omega}(X - Z)||_F^2 + \lambda\|Z\|_*. \tag{3.28}$$

For each given maximum rank, MMMF produces an estimate by doing further shrinkage with its quadratic regularization. Soft-Impute performs rank reduction and shrinkage at the same time, in one smooth convex operation. The following theorem shows that this one-dimensional Soft-Impute family lies exactly in the two-dimensional MMMF family.

Theorem 8. *Let X be $m \times n$ with observed entries indexed by Ω.*

1. *Let $r' = \min(m, n)$. Then the solutions to (3.27) and (3.28) coincide for all $\lambda \geq 0$.*

2. *Suppose $\hat{Z}^*$ is a solution to (3.28) for $\lambda^* > 0$, and let r^* be its rank. Then for any solution $\hat{U}$, $\hat{V}$ to (3.27) with $r' = r^*$ and $\lambda = \lambda^*$, $\hat{U}\hat{V}^T$ is a solution to (3.28). The SVD factorization of $\hat{Z}^*$ provides one such solution to (3.27). This implies that the solution space of (3.28) is contained in that of (3.27).*

Remarks:

1. Part 1 of this theorem appears in a slightly different form in Srebro, Rennie & Jaakkola (2005).

2. In part 1, we could use $r' > \min(m, n)$ and get the same equivalence. While this might seem unnecessary, there may be computational advantages; searching over a

bigger space might protect against local minima. Likewise in part 2, we could use $r' > r^*$ and achieve the same equivalence. In either case, no matter what r' we use, the solution matrices $\hat{U}$ and $\hat{V}$ have the same rank as $\hat{Z}$.

3. Let $\hat{Z}(\lambda)$ be a solution to (3.28) at λ. We conjecture that rank $[\hat{Z}(\lambda)]$ is monotone non-increasing in λ. If this is the case, then Theorem 8, part 2 can be further strengthened to say that for all $\lambda \geq \lambda^*$ and $r' = r^*$ the solutions of (3.27) coincide with that of (3.28).

The MMMF criterion (3.27) defines a two-dimensional family of models indexed by (r', λ), while the SOFT-IMPUTE criterion (3.28) defines a one-dimensional family. In light of Theorem 8, this family is a special path in the two-dimensional grid of solutions $[\hat{U}_{(r',\lambda)}, \hat{V}_{(r',\lambda)}]$. Figure 3.1 depicts the situation. Any MMMF model at parameter combinations above the

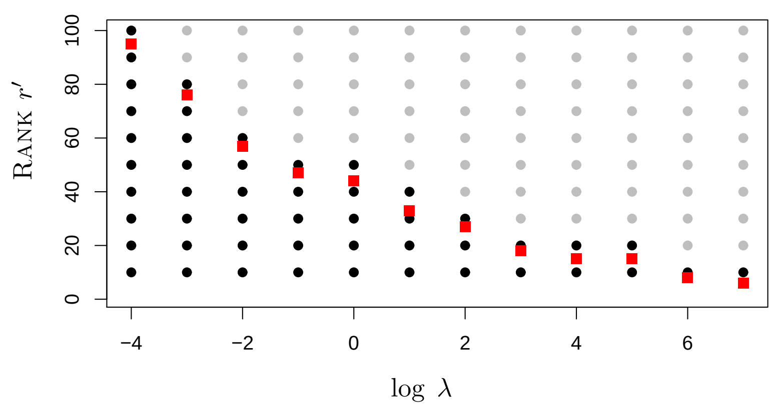

Figure 3.1: Comparison of the parameter space for MMMF (grey and black points), and SOFT-IMPUTE (red squares) for a simple example. Since all MMMF solutions with parameters above the red squares are identical to the SOFT-IMPUTE solutions at the red squares, all the grey points are redundant.

red squares are redundant, since their fit is the same at the red square. However, in practice the red squares are not known to MMMF, nor is the actual rank of the solution. Further orthogonalization of $\hat{U}$ and $\hat{V}$ would be required to reveal the rank, which would only be approximate (depending on the convergence criterion of the MMMF algorithm).

Despite the equivalence of (3.27) and (3.28) when $r' = \min(m, n)$, the criteria are quite different. While (3.28) is a convex optimization problem in Z, (3.27) is a non-convex problem in the variables U, V and has possibly several local minima; see also Abernethy et al. (2009). It has been observed empirically and theoretically (Burer & Monteiro 2005, Rennie & Srebro 2005) that bi-convex methods used in the optimization of (3.27) can get stuck in sub-optimal local minima for a small value of r' or a poorly chosen starting point. For a large number of factors r' and large dimensions m, n the computational cost may be quite high (See also experimental studies in Section 3.9.2).

Criterion (3.28) is convex in Z for every value of λ, and it outputs the solution $\hat{Z}$ in the form of its soft-thresholded SVD, implying that the "factors" U, V are already orthogonal and the rank is known.

MMMF has two different tuning parameters r' and λ, both of which are related to the rank or spectral properties of the matrices U, V. Soft-Impute has only one tuning parameter λ. The presence of two tuning parameters is problematic:

- It results in a significant increase in computational burden, since for every given value of r', one needs to compute an entire system of solutions by varying λ (see Section 3.9 for illustrations).

- In practice when neither the optimal values of r' and λ are known, a two-dimensional search (for example, by cross validation) is required to select suitable values.

Further discussions and connections between the tuning parameters and spectral properties of the matrices can be found in Burer & Monteiro (2005) and Abernethy et al. (2009).

The proof of Theorem 8 requires a lemma.

Lemma 11. *For any matrix Z, the following holds:*

$$||Z||_* = \min_{U,V:\, Z=UV^T} \tfrac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right). \tag{3.29}$$

If $\operatorname{rank}(Z) = k \leq \min\{m, n\}$, *then the minimum above is attained at a factor decomposition* $Z = U_{m\times k}V_{n\times k}^T$.

Note that in the decomposition $Z = UV^T$ in (3.29) there is no constraint on the number of columns r of the factor matrices $U_{m\times r}$ and $V_{n\times r}$. Lemma 11 is stronger than similar results appearing in Rennie & Srebro (2005) and Abernethy et al. (2009) which establish (3.29) for $r = \min\{m, n\}$—we give a tighter estimate of the rank k of the underlying matrices. The proof is given in Appendix 3.A.5.

3.8.1 Proof of Theorem 8

Part 1. For $r = \min(m, n)$, any matrix $Z_{m\times n}$ can be written in the form of $Z = UV^T$. The criterion (3.27) can be written as

$$\begin{aligned} &\min_{U,V} \quad \tfrac{1}{2}||P_\Omega(X - UV^T)||_F^2 + \tfrac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) && (3.30)\\ = &\min_{U,V} \quad \tfrac{1}{2}||P_\Omega(X - UV^T)||_F^2 + \lambda\|UV^T\|_* \qquad \text{(by Lemma 11)} && \\ = &\min_{Z} \quad \tfrac{1}{2}||P_\Omega(X - Z)||_F^2 + \lambda\|Z\|_*. && (3.31) \end{aligned}$$

The equivalence of the criteria in (3.30) and (3.31) completes the proof of part 1.

Part 2. Note that if we know that the solution $\hat{Z}^*$ to (3.28) with $\lambda = \lambda^*$ has rank r^*, then $\hat{Z}^*$ also solves

$$\min_{Z,\, \operatorname{rank}(Z)=r^*} \quad \tfrac{1}{2}||P_\Omega(X - Z)||_F^2 \quad + \lambda\|Z\|_*.$$

We now repeat the steps (3.30)—(3.31), restricting the rank r' of U and V to be $r' = r^*$, and the result follows.

3.9 Numerical Experiments and Comparisons

In this section we study the performance of Soft-Impute, its post-processed variants, and Hard-Impute for noisy matrix completion problems. The examples assert our claim that the matrix reconstruction criterion (3.4) (Cai et al. 2010) is too rigid if one seeks good predictive models. We include the related procedures of Rennie & Srebro (2005) and Keshavan et al. (2009) in our comparisons.

The reconstruction algorithm OptSpace, described in Keshavan et al. (2009) considers criterion (3.1) (in the presence of noise). It uses the representation $Z = USV'$ (which need not correspond to the SVD). OptSpace alternates between estimating S and U, V (in a Grassmann manifold) for computing a rank-r decomposition $\hat{Z} = \hat{U}\hat{S}\hat{V}'$. It starts with a sparse SVD on a *clean* version of the observed matrix $P_\Omega(X)$. This is similar to the formulation of MMMF (3.27) as detailed in Section 3.8, without the squared Frobenius norm regularization on the components U, V.

To summarize, we study the following methods:

1. Soft-Impute–Algorithm 2; denoted by SoftImp in the figure labels.

2. Soft-Impute+–post-processing on the output of Soft-Impute, as in Section 3.7; denoted by SoftImp+ in the figure labels.

3. Hard-Impute–Algorithm 3, starting with the output of Soft-Impute+. We denote this by HardImp in the figure labels.

4. SVT–algorithm by Cai et al. (2010);

5. OptSpace–reconstruction algorithm by Keshavan et al. (2009);

6. MMMF–algorithm for (3.6) as in Rennie & Srebro (2005).

In all our simulation studies we use the underlying model $Z_{m\times n} = U_{m\times r}V'_{r\times n} + \varepsilon$, where U and V are random matrices with standard normal Gaussian entries, and ε is i.i.d. Gaussian.

Ω is uniformly random over the indices of the matrix with $p\%$ percent of missing entries. These are the models under which the coherence conditions hold true for the matrix completion problem to be meaningful (Candès & Tao 2009, Keshavan et al. 2009). The signal to noise ratio for the model and the test-error (standardized) are defined as

$$\text{SNR} = \sqrt{\frac{\text{var}(UV')}{\text{var}(\varepsilon)}}; \quad \text{Test Error} = \frac{\|P_\Omega^\perp(UV' - \hat{Z})\|_F^2}{\|P_\Omega^\perp(UV')\|_F^2}.$$

Training error (standardized) is defined as

$$\text{Training Error} = \frac{\|P_\Omega(Z - \hat{Z})\|_F^2}{\|P_\Omega(Z)\|_F^2},$$

the fraction of the error explained on the observed entries by the estimate relative to a zero estimate.

Figures 3.2, 3.3 and 3.4 show training and test error for all of the algorithms mentioned above—both as a function of nuclear norm and rank—for the three problem instances. The results displayed in the figures are averaged over 50 simulations, and also show one-standard-error bands (hardly visible). In all examples $(m, n) = (100, 100)$. For MMMF we use $r' = \min(m, n) = 100$, the number of columns in U and V. The performance of MMMF is displayed only in the plots with the nuclear norm along the horizontal axis, since the algorithm does not deliver a precise rank. SNR, true rank and percentage of missing entries are indicated in the figures. There is a unique correspondence between λ and nuclear norm. The plots versus rank indicate how effective the nuclear norm is as a rank approximation—that is whether it recovers the true rank while minimizing prediction error.

For routines not our own we use the MATLAB code as supplied on webpages by the authors. For SVT second author of Cai et al. (2010), for OptSpace third author of Keshavan et al. (2009), and for MMMF first author of Rennie & Srebro (2005).

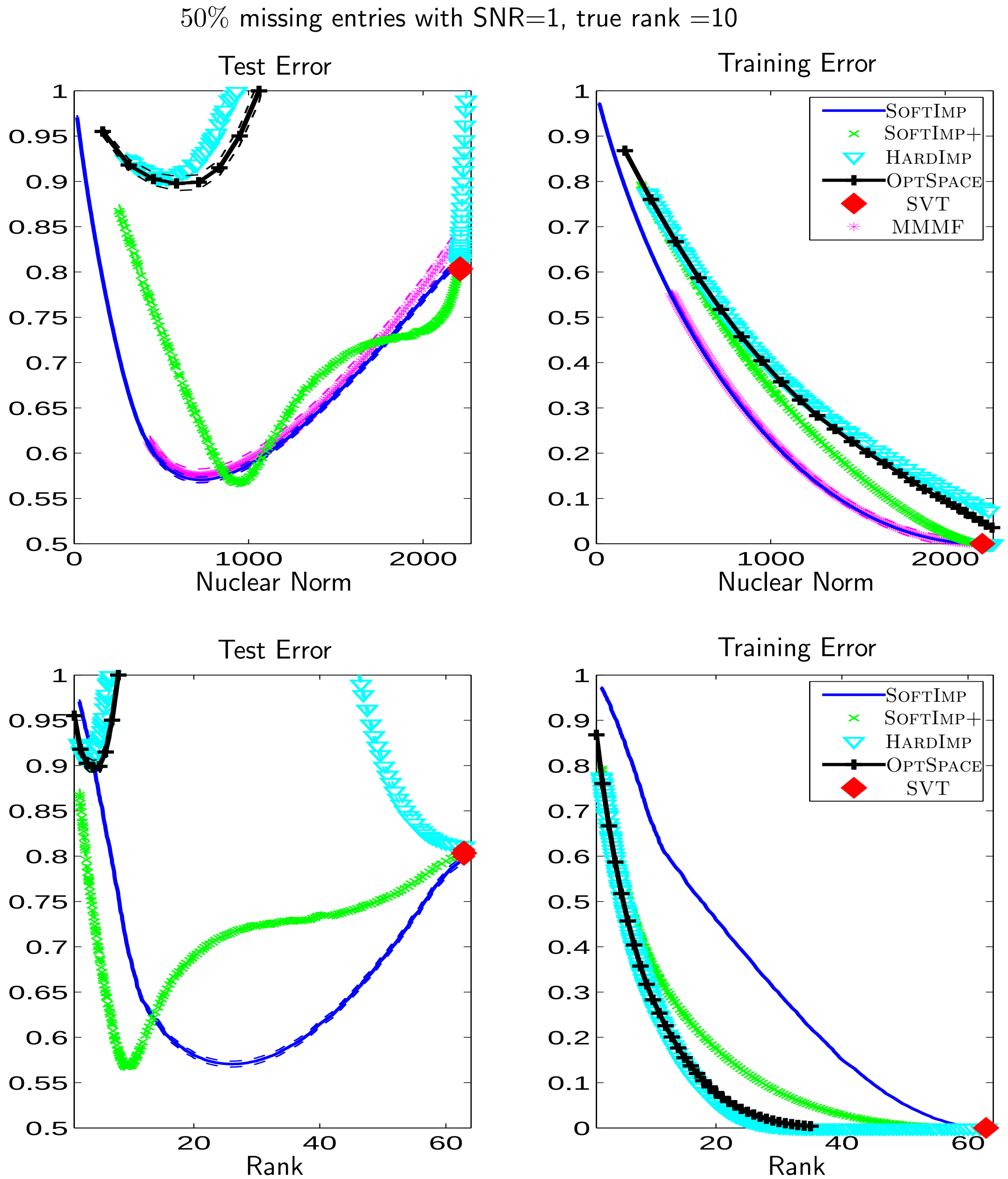

Figure 3.2: Figure showing training/test error profiles for Soft-Impute+, Soft-Impute, Hard-Impute OptSpace, SVT, MMMF (with full rank 100). Values of test error larger than one are not shown in the figure. OptSpace is evaluated for ranks ≤ 30.

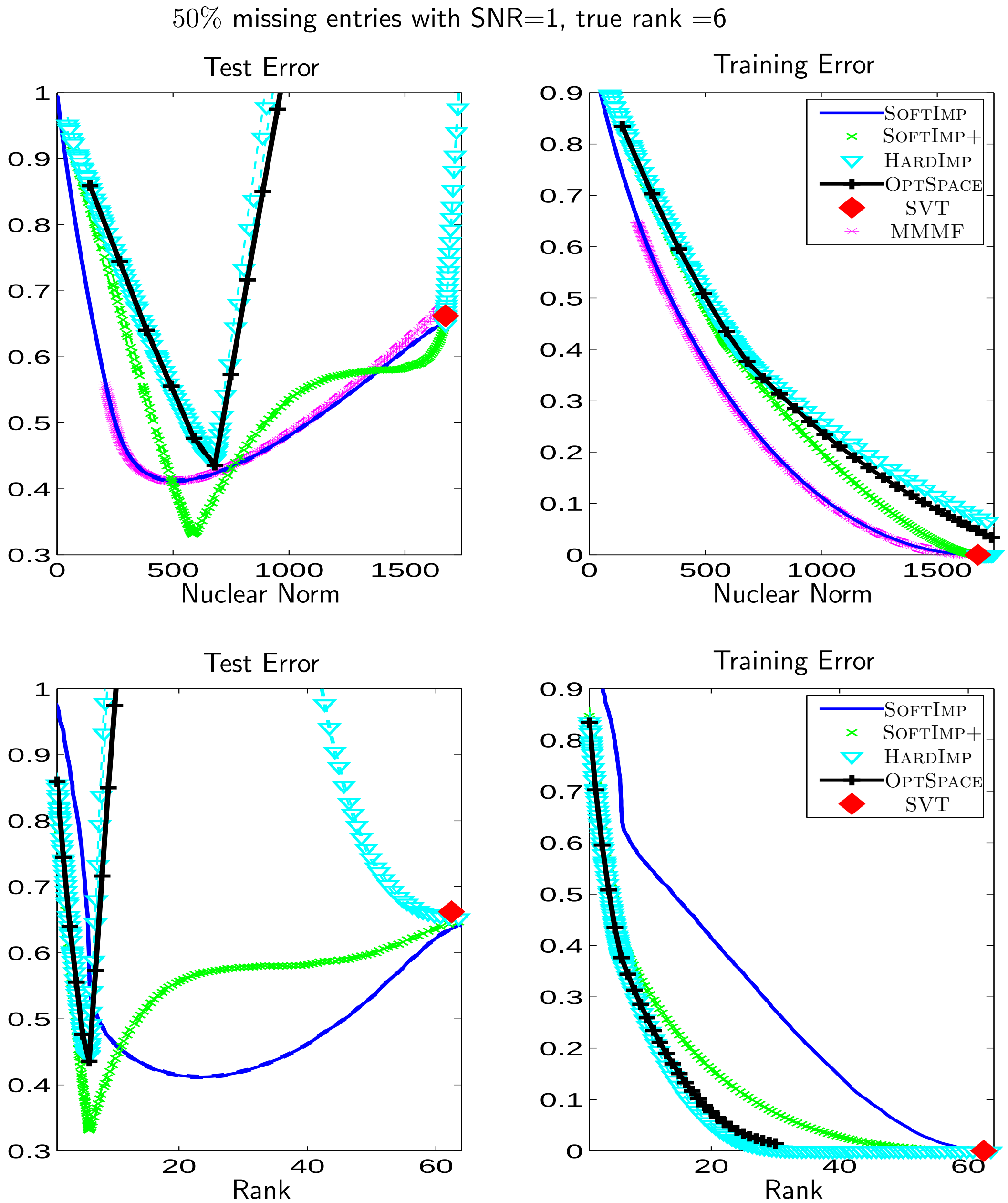

Figure 3.3: Soft-Impute+ has the best prediction error, closely followed by Soft-Impute and MMMF. Both Hard-Impute and OptSpace have poor prediction error apart from near the true rank. OptSpace is evaluated for a series of ranks ≤ 35. SVT has poor predictiion error with rank of the solution being > 60.

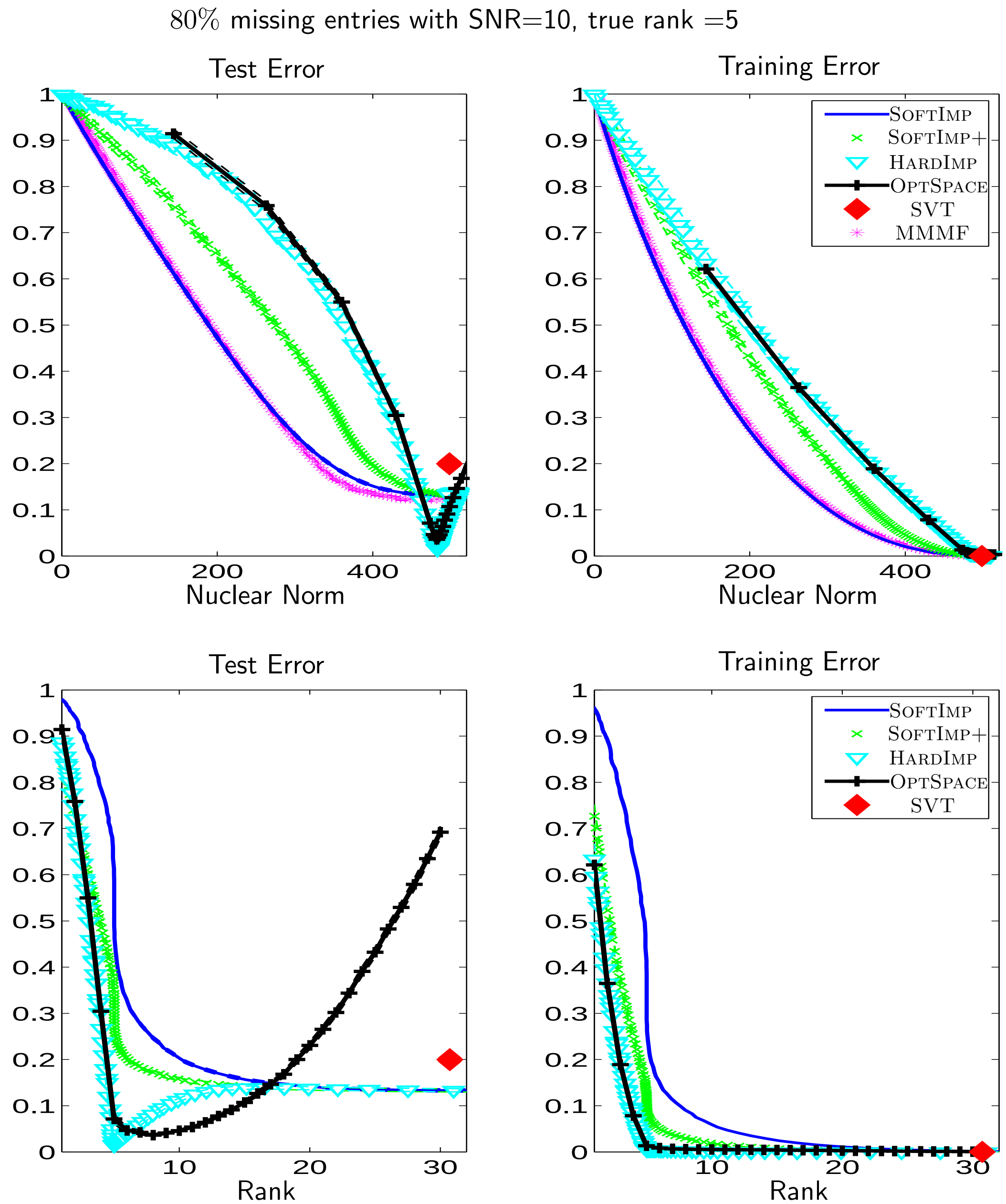

Figure 3.4: With low noise, the performance of HARD-IMPUTE improves and has the best prediction error, followed by OPTSPACE. SVT did not converge within 1000 iterations, and the solution seems to be different from the limiting solution of SOFT-IMPUTE

3.9.1 Observations

The captions of each of Figures 3.2–3.4 detail the results, which we summarize here. For the first two figures, the noise is quite high with SNR= 1, and 50% of the entries are missing. In Figure 3.2 the true rank is 10, while in Figure 3.3 it is 6. SOFT-IMPUTE, MMMF and SOFT-IMPUTE+ have the best prediction performance, while SOFT-IMPUTE+ is better at estimating the correct rank. The other procedures perform poorly here, although OPTSPACE improves somewhat in Figure 3.3. SVT has very poor prediction error, suggesting once again that exactly fitting the training data is far too rigid. SOFT-IMPUTE+ has the best performance in Figure 3.3 (smaller rank—more aggressive fitting), and HARD-IMPUTE starts recovering here. In both figures the training error for SOFT-IMPUTE (and hence MMMF) wins as a function of nuclear norm (as it must, by construction), but the more aggressive fitters SOFT-IMPUTE+ and HARD-IMPUTE have better training error as a function of rank.

Though the nuclear norm is often viewed as a surrogate for the rank of a matrix, we see in these examples that it can provide a superior mechanism for regularization. This is similar to the performance of LASSO in the context of regression. Although the LASSO penalty can be viewed as a convex surrogate for the ℓ_0 penalty in model selection, its ℓ_1 penalty provides a smoother and often better basis for regularization.

In Figure 3.4 with SNR= 10 the noise is relatively small compared to the other two cases. The true underlying rank is 5, but the proportion of missing entries is much higher at eighty percent. Test errors of both SOFT-IMPUTE+ and SOFT-IMPUTE are found to decrease till a large nuclear norm after which they become roughly the same, suggesting no further impact of regularization. MMMF has slightly better test error than SOFT-IMPUTE around a nuclear norm of 350, while in theory they should be identical. Notice, however, that the training error is slightly worse (everywhere), suggesting that MMMF is sometimes trapped in local minima. The fact that this slightly underfit solution does better in test error is a quirk of this particular example. OptSpace performs well in this high-SNR example, achieving a sharp minima at the true rank of the matrix. Hard-Impute performs the best in this example. The better performance of both OptSpace and Hard-Impute over Soft-Impute can be attributed both to the low-rank truth and the high SNR. This is reminiscent of the better predictive performance of best-subset or concave penalized regression often seen over lasso in setups where the underlying model is very sparse (Friedman 2008).

3.9.2 Comparison with Fast MMMF

In this section we compare Soft-Impute with MMMF (Rennie & Srebro 2005) in terms of computational efficiency. We also examine the consequences of two regularization parameters (r', λ) for MMMF over one for Soft-Impute.

Rennie & Srebro (2005) describes a fast algorithm based on conjugate-gradient descent for minimization of the MMMF criterion (3.6). With (3.6) being non-convex, it is hard to provide theoretical optimality guarantees for the algorithm for arbitrary r', λ—that is, what type of solution it converges to or how far it is from the global minimizer.

In Table 3.1 we summarize the performance results of the two algorithms. For both Soft-Impute and MMMF we consider a equi-spaced grid of 150 $\lambda \in [\lambda_{\min}, \lambda_{\max}]$, with $\lambda_{\min}$ corresponding to a full-rank solution of Soft-Impute and $\lambda_{\max}$ the zero solution. For MMMF, three different values of r' were used, and for each $(\hat{U}, \hat{V})$ were solved for over the grid of λ values. A separate held-out validation set with twenty percent of the missing entries sampled from $\Omega^{\perp}$ were used to train the tuning parameter λ (for each value of r') for MMMF and Soft-Impute. Finally we evaluate the standardized prediction errors on a test set consisting of the remaining eighty percent of the missing entries in $\Omega^{\perp}$. In all cases we report the training errors and test errors on the optimally tuned λ. Soft-Impute was run till a tolerance of 10^{-4} was achieved (fraction of decrease of objective value). Likewise for MMMF we set the tolerance of the conjugate gradient method to 10^{-4}.

Data	Method (rank)	Test error	Training error	Time (secs)
$(m, n) = (10^2, 10^2)$	SOFT-IMPUTE (39)	0.7238 (0.0027)	0.0720	4.1745
$\|\Omega\| = 5 \times 10^3 (50\%)$	MMMF (20)	0.7318 (0.0031)	0.2875	48.1090
SNR= 3	MMMF (60)	0.7236 (0.0027)	0.1730	62.0230
rank (R)= 30	MMMF (100)	0.7237 (0.0027)	0.1784	96.2750
$(m, n) = (10^2, 10^2)$	SOFT-IMPUTE (37)	0.5877 (0.0047)	0.0017	4.0976
$\|\Omega\| = 2 \times 10^3 (20\%)$	MMMF (20)	0.5807 (0.0049)	0.0186	53.7533
SNR= 10	MMMF (60)	0.5823 (0.0049)	0.0197	62.0230
rank(R)= 10	MMMF (100)	0.5823 (0.0049)	0.0197	84.0375
$(m, n) = (10^2, 10^2)$	SOFT-IMPUTE (59)	0.6008 (0.0028)	0.0086	3.8447
$\|\Omega\| = 8 \times 10^3 (80\%)$	MMMF (20)	0.6880 (0.0029)	0.4037	33.8685
SNR= 10	MMMF (60)	0.5999 (0.0028)	0.0275	57.3488
rank(R)= 45	MMMF (100)	0.5999 (0.0028)	0.0275	89.4525

Table 3.1: Performances of SOFT-IMPUTE and MMMF for different problem instances, in terms of test error (with standard errors in parentheses), training error and times for learning the models. SOFT-IMPUTE,"rank" denotes the rank of the recovered matrix, at the optimally chosen value of λ. For the MMMF, "rank" indicates the value of r' in $U_{m\times r'}, V_{n\times r'}$. Results are averaged over 50 simulations.

In Table 3.1, for every algorithm total time indicates the time required for evaluating solutions over the entire grid of λ values. In these examples, we used direct SVD factorization based methods for the SVD computation, since the size of the problems were quite small. In all these examples we observe that SOFT-IMPUTE performs very favorably in terms of total times. For MMMF the time to train the models increase with increasing rank r'; and in case the underlying matrix has rank which is larger than r', the computational cost will be large in order to get competitive predictive accuracy. This point is supported in the examples of Table 3.1. It is important to note that, the prediction error of SOFT-IMPUTE as obtained on the validation set is actually within standard error of the best prediction error produced by all the MMMF models.

In addition we also performed some medium-scale examples increasing the dimensions of the matrices. To make comparisons fair, SOFT-IMPUTE made use of *direct* SVD computations (in MATLAB) instead of iterative algorithms exploiting the specialized *Sparse+Low-Rank* structure (3.22). We report our findings on one such simulation example:

- For $(m, n) = (2000, 1000)$, $|\Omega|/(m \cdot n) = 0.2$, rank = 500 and SNR=10; SOFT-IMPUTE takes 1.29 hours to compute solutions on a grid of 100 λ values. The test error on the validation set and training error are 0.9630 and 0.4375 with the recovered solution having a rank of 225.

  For the same problem, MMMF with $r' = 200$ takes 6.67 hours returning a solution with test-error 0.9678 and training error 0.6624. With $r' = 400$ it takes 12.89 hrs with test and training errors 0.9659 and 0.6564 respectively.

We will like to note that DeCoste (2006) proposed an efficient implementation of MMMF via an ensemble based approach, which is quite different in spirit from the batch optimization algorithms we are studying in this paper. Hence we do not compare it with SOFT-IMPUTE.

3.9.3 Comparison with Nesterov's Accelerated Gradient Method

Ji & Ye (2009) proposed a first-order algorithm based on Nesterov's acceleration scheme (Nesterov 2007), for nuclear norm minimization for a generic multi-task learning problem (Argyriou et al. 2008, Argyriou et al. 2007). Their algorithm (Liu et al. 2009, Ji & Ye 2009) can be adapted to the SOFT-IMPUTE problem (3.10); hereafter we refer to it as NESTEROV. It requires one to compute the SVD of a dense matrix having the dimensions of X, which makes it prohibitive for large-scale problems. We instead would make use of the structure (3.22) for the SVD computation, a special characteristic of matrix completion which is not present in a generic multi-task learning problem. Here we compare the performances of SOFT-IMPUTE and NESTEROV on small scale examples, where direct SVDs can be computed easily.

Since both algorithms solve the same criterion, the quality of the solutions—objective values, training and test errors—will be the same (within tolerance). We hence compare their performances based on the times taken by the algorithms to converge to the optimal solution of (3.10) on a grid of values of λ. Both algorithms compute a path of solutions using warm starts. Results are shown in Figure 3.5, for four different scenarios described in Table 3.2.

Example	(m, n)	$\lvert\Omega\rvert/(m \cdot n)$	Rank	Test Error
i	$(100, 100)$	0.5	5	0.4757
ii	$(100, 100)$	0.2	5	0.0668
iii	$(100, 100)$	0.8	5	0.0022
iv	$(1000, 500)$	0.5	30	0.0028

Table 3.2: Four different examples used for timing comparisons of Soft-Impute and (accelerated) Nesterov. In all cases the SNR= 10.

Figure 3.5 shows the time to convergence for the two algorithms. Their respective number of iterations are not comparable. This is because Nesterov uses a line-search to compute an adaptive step-size (approximate the Lipschitz constant) at every iteration, whereas Soft-Impute does not.

Soft-Impute has a rate of convergence given by Theorem 7, which for large k is worse than the accelerated version Nesterov with rate $O(1/k^2)$. However, timing comparisons in Figure 3.5 show that soft-impute performs very favorably. We do not know the exact reason behind this, but mention some possibilities. Firstly the rates are *worst case* convergence rates. On particular problem instances of the form (3.10), the rates of convergence in *practice* of soft-impute and Nesterov may be quite similar. Since Ji & Ye (2009) uses an adaptive step-size strategy, the choice of a step-size may be time consuming. soft-impute on the other hand, uses a constant step size.

Additionally, it appears that the use of the *momentum* term in Nesterov affects the *Sparse+Low-rank* decomposition (3.22). This may prevent the algorithm to be adapted for solving large problems, due to costly SVD computations.

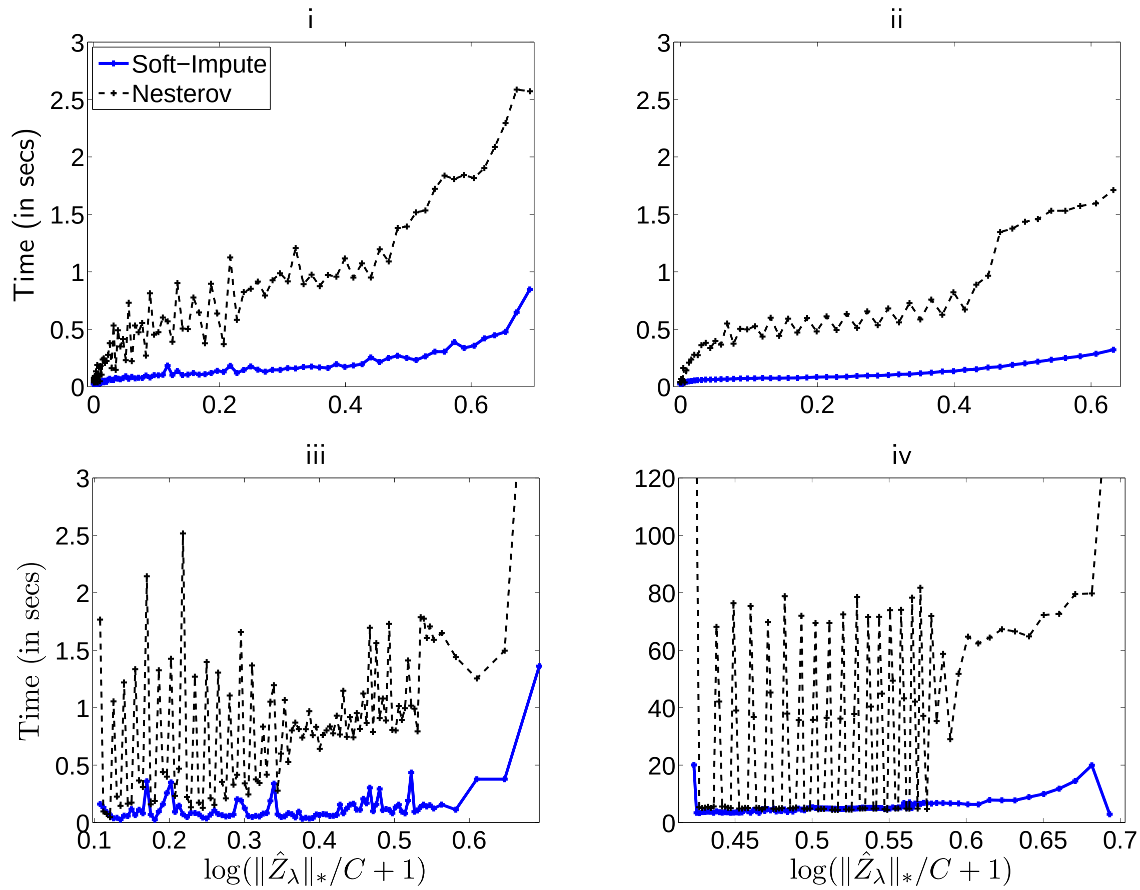

Figure 3.5: Timing comparisons of SOFT-IMPUTE and (accelerated) NESTEROV over a grid of λ values for examples i–iv. The horizontal axis corresponds to the standardized nuclear norm, with $C = \max_\lambda \|\hat{Z}_\lambda\|_*$. Results are averaged over 10 simulations.

3.9.4 Large Scale Simulations for Soft-Impute

Table 3.3 reports the performance of Soft-Impute on some large-scale problems. All computations are performed in MATLAB and the MATLAB implementation of PROPACK is used. Data input, access and transfer in MATLAB take a sizable portion of the total computational time, given the size of these problems. However, the main computational bottle neck in our algorithm is the structured SVD computation. In order to focus more on the essential computational task, Table 3.3 displays the total time required to perform the SVD computations over all iterations of the algorithm. Note that for all the examples considered in Table 3.3, the implementations of algorithms Nesterov (Liu et al. 2009, Ji & Ye 2009) and MMMF (Rennie & Srebro 2005) are prohibitively expensive both in terms of computational time and memory requirements, and hence could not be run. We used the value $\lambda = ||P_\Omega(X)||_2/K$ with Soft-Impute, with $K = 1.5$ for all examples but the last, where $K = 2$. $\lambda_0 = ||P_\Omega(X)||_2$ is the largest singular value of the input matrix X (padded with zeros); this is the smallest value of λ for which $\mathrm{S}_{\lambda_0}(P_\Omega(X)) = 0$ in the first iteration of Soft-Impute (Section 3.3).

The prediction performance is awful for all but one of the models, because in most cases the fraction of observed data is very small. These simulations were mainly to show the computational capabilities of Soft-Impute on very large problems.

3.10 Application to the Netflix Data Set

The Netflix training data consists of the ratings of 17,770 movies by 480,189 Netflix customers. The resulting data matrix is extremely sparse, with 100,480,507 or 1% of the entries observed. The task was to predict the unseen ratings for a qualifying set and a test set of about 1.4 million ratings each, with the true ratings in these data sets held in secret by Netflix. A probe set of about 1.4 million ratings was distributed to participants, for

(m, n)	$\|\Omega\|$	$\|\Omega\|/(m \cdot n)$ $\times 100\%$	Recovered rank	Time (mins)	Test error	Training error
$(10^4, 10^4)$	10^5	0.1	40^*	0.2754	0.9946	0.6160
$(10^4, 10^4)$	10^5	0.1	40^*	0.3770	0.9959	0.6217
$(10^4, 10^4)$	10^5	0.1	50^*	0.4292	0.9962	0.3862
$(10^4, 10^4)$	10^6	1.0	5	1.6664	0.6930	0.6600
$(10^5, 10^5)$	5×10^6	0.05	40^*	17.2518	0.9887	0.8156
$(10^5, 10^5)$	10^7	0.1	5	20.3142	0.9803	0.9761
$(10^6, 10^5)$	10^8	0.1	5	259.9620	0.9913	0.9901
$(10^6, 10^6)$	10^7	0.001	20^*	99.6780	0.9998	0.5834

Table 3.3: Performance of SOFT-IMPUTE on different problem instances. All models are generated with SNR=10 and underlying rank=5. Recovered rank is the rank of the solution matrix $\hat{Z}$ at the value of λ used in (3.10). Those with stars reached the "maximum rank" threshold, and option in our algorithm. Convergence criterion is taken as "fraction of improvement of objective value" less than 10^{-4} or a maximum of 15 iterations for the last four examples. All implementations are done in MATLAB including the MATLAB implementation of PROPACK on a Intel Xeon Linux 3GHz processor.

calibration purposes. The movies and customers in the qualifying, test and probe sets are all subsets of those in the training set.

The ratings are integers from 1 (poor) to 5 (best). Netflix's own algorithm has an RMSE of 0.9525, and the contest goal was to improve this by 10%, or an RMSE of 0.8572 or better. The contest ran for about 3 years, and the winning team was "Bellkor's Pragmatic Chaos", a merger of three earlier teams (see `http://www.netflixprize.com/` for details). They claimed the grand prize of $1M on September 21, 2009.

Many of the competitive algorithms build on a regularized low-rank factor model similar to (3.6) using randomization schemes like mini-batch, stochastic gradient descent or sub-sampling to reduce the computational cost over making several passes over the entire data-set (Salakhutdinov et al. 2007, Bell & Koren. 2007, Takacs et al. 2009, see,for example). In this paper, our focus is not on using randomized or sub-sampling schemes.

Here we demonstrate that our nuclear-norm regularization algorithm can be applied in batch mode on the entire Netflix training set with a reasonable computation time. We note however that the conditions under which the nuclear-norm regularization is theoretically

λ	Rank	Time (hrs)	RMSE
$\lambda_0/250$	42	1.36	0.9622
$\lambda_0/300$	66	2.21	0.9572
$\lambda_0/500$	81	2.83	0.9543
$\lambda_0/600$	95	3.27	0.9497

Table 3.4: Results of applying Soft-Impute to the Netflix data. $\lambda_0 = ||P_\Omega(X)||_2$; see Section 3.9.4. The computations were done on a Intel Xeon Linux 3GHz processor; timings are reported based on MATLAB implementations of PROPACK and our algorithm. RMSE is root-mean squared error, as defined in the text.

meaningful (Candès & Tao 2009, Srebro, Alon & Jaakkola 2005) are not met on the Netflix data set.

We applied Soft-Impute to the training data matrix and then performed a least-squares unshrinking on the singular values with the singular vectors and the training data row and column means as the bases. The latter was performed on a data-set of size 10^5 randomly drawn from the probe set. The prediction error (RMSE) is obtained on a left out portion of the probe set. Table 3.4 reports the performance of the procedure for different choices of the tuning parameter λ (and the corresponding rank); times indicate the total time taken for the SVD computations over all iterations. A maximum of 10 iterations were performed for each of the examples. Again, these results are not competitive with those of the competition leaders, but rather demonstrate the feasibility of applying Soft-Impute to such a large data set.

3.A Appendix to Chapter 3

We begin with the proof of Lemma 6.

3.A.1 Proof of Lemma 6

Proof. Let $Z = \tilde{U}_{m\times n}\tilde{D}_{n\times n}\tilde{V}'_{n\times n}$ be the SVD of Z. Assume without loss of generality, $m \geq n$. We will explicitly evaluate the closed form solution of the problem (3.8). Note that

$$\tfrac{1}{2}\|Z - W\|_F^2 + \lambda\|Z\|_* = \tfrac{1}{2}\left\{\|W\|_F^2 - 2\sum_{i=1}^{n}\tilde{d}_i\tilde{u}_i'W\tilde{v}_i + \sum_{i=1}^{n}\tilde{d}_i^2\right\} + \lambda\sum_{i=1}^{n}\tilde{d}_i \tag{3.32}$$

where

$$\tilde{D} = \operatorname{diag}\left[\tilde{d}_1,\ldots,\tilde{d}_n\right],\ \ \tilde{U} = [\tilde{u}_1,\ldots,\tilde{u}_n]\,,\ \ \tilde{V} = [\tilde{v}_1,\ldots,\tilde{v}_n]\,.$$

Minimizing (3.32) is equivalent to minimizing

$$-2\sum_{i=1}^{n}\tilde{d}_i\tilde{u}_i'W\tilde{v}_i + \sum_{i=1}^{n}\tilde{d}_i^2 + \sum_{i=1}^{n}2\lambda\tilde{d}_i;\ \ \text{w.r.t. } (\tilde{u}_i,\tilde{v}_i,\tilde{d}_i),\ i = 1,\ldots,n,$$

under the constraints $\tilde{U}'\tilde{U} = I_n$, $\tilde{V}'\tilde{V} = I_n$ and $\tilde{d}_i \geq 0\ \ \forall i$.

Observe the above is equivalent to minimizing (w.r.t. $\tilde{U},\tilde{V}$) the function $Q(\tilde{U},\tilde{V})$:

$$Q(\tilde{U},\tilde{V}) = \min_{\tilde{D}\geq 0}\ \ \tfrac{1}{2}\left\{-2\sum_{i=1}^{n}\tilde{d}_i\tilde{u}_i'W\tilde{v}_i + \sum_{i=1}^{n}\tilde{d}_i^2\right\} + \lambda\sum_{i=1}^{n}\tilde{d}_i. \tag{3.33}$$

Since the objective (3.33) to be minimized w.r.t. $\tilde{D}$, is separable in $\tilde{d}_i, i = 1,\ldots,n$; it suffices to minimize it w.r.t. each $\tilde{d}_i$ separately.

The problem

$$\operatorname*{minimize}_{\tilde{d}_i\geq 0}\ \tfrac{1}{2}\left\{-2\tilde{d}_i\tilde{u}_i'W\tilde{v}_i + \tilde{d}_i^2\right\} + \lambda\tilde{d}_i \tag{3.34}$$

can be solved looking at the stationary conditions of the function using its sub-gradient (Nesterov 2003). The solution of the above problem is given by $S_\lambda(\tilde{u}_i'W\tilde{v}_i) = (\tilde{u}_i'W\tilde{v}_i - \lambda)_+$, the soft-thresholding of $\tilde{u}_i'W\tilde{v}_i$ (without loss of generality, we can take $\tilde{u}_i'W\tilde{v}_i$ to be non-negative). More generally the soft-thresholding operator (Friedman, Hastie, Hoefling &

Tibshirani 2007, Hastie et al. 2009) is given by $S_\lambda(x) = \text{sgn}(x)(|x| - \lambda)_+$. See Friedman, Hastie, Hoefling & Tibshirani (2007) for more elaborate discussions on how the soft-thresholding operator arises in univariate penalized least-squares problems with the ℓ_1 penalization.

Plugging the values of optimal $\tilde{d}_i, i = 1, \ldots, n$; obtained from (3.34) in (3.33) we get

$$Q(\tilde{U}, \tilde{V}) = \tfrac{1}{2}\left\{ \|W\|_F^2 - 2\sum_{i=1}^{n} (\tilde{u}_i' W \tilde{v}_i - \lambda)_+ (\tilde{u}_i' W \tilde{v}_i - \lambda) + (\tilde{u}_i' X \tilde{v}_i - \lambda)_+^2 \right\}. \tag{3.35}$$

Minimizing $Q(\tilde{U}, \tilde{V})$ w.r.t. $(\tilde{U}, \tilde{V})$ is equivalent to maximizing

$$\sum_{i=1}^{n} \left\{ 2(\tilde{u}_i' W \tilde{v}_i - \lambda)_+ (\tilde{u}_i' W \tilde{v}_i - \lambda) - (\tilde{u}_i' W \tilde{v}_i - \lambda)_+^2 \right\} = \sum_{\tilde{u}_i' W \tilde{v}_i > \lambda} (\tilde{u}_i' W \tilde{v}_i - \lambda)^2. \tag{3.36}$$

It is a standard fact that for every i the problem

$$\underset{\|u\|_2^2 \leq 1, \|v\|_2^2 \leq 1}{\text{maximize}} \; u' W v; \;\; \text{such that} \;\; u \perp \{\hat{u}_1, \ldots, \hat{u}_{i-1}\}, \; v \perp \{\hat{v}_1, \ldots, \hat{v}_{i-1}\}$$

is solved by $\hat{u}_i, \hat{v}_i$, the left and right singular vectors of the matrix W corresponding to its i^{th} largest singular value. The maximum value equals the singular value. It is easy to see that maximizing the expression to the right of (3.36) wrt $(u_i, v_i), i = 1, \ldots, n$ is equivalent to maximizing the individual terms $\tilde{u}_i' W \tilde{v}_i$. If $r(\lambda)$ denotes the number of singular values of W larger than λ then the $(\tilde{u}_i, \tilde{v}_i), i = 1, \ldots$ that maximize the expression (3.36) correspond to $\left[u_1, \ldots, u_{r(\lambda)}\right]$ and $\left[v_1, \ldots, v_{r(\lambda)}\right]$; the $r(\lambda)$ left and right singular vectors of W corresponding to the largest singular values. From (3.34) the optimal $\tilde{D} = \text{diag}\left[\tilde{d}_1, \ldots, \tilde{d}_n\right]$ is given by $D_\lambda = \text{diag}\left[(d_1 - \lambda)_+, \ldots, (d_n - \lambda)_+\right]$.

Since the rank of W is r, the minimizer $\hat{Z}$ of (3.8) is given by $UD_\lambda V'$ as in (3.9). $\square$

Remark 1. *For a more general spectral regularization of the form $\lambda \sum_i \mathbf{p}(\gamma_i(Z))$ (as compared to $\sum_i \lambda\gamma_i(Z)$ used above) the optimization problem (3.34) will be modified accordingly.*

The solution of the resultant univariate minimization problem will be given by $S^{\mathbf{p}}_{\lambda}(\tilde{u}_i' W \tilde{v}_i)$ *for some generalized "thresholding operator"* $S^{\mathbf{p}}_{\lambda}(\cdot)$*, where*

$$S^{\mathbf{p}}_{\lambda}(\tilde{u}_i' W \tilde{v}_i) = \arg\min_{\tilde{d}_i \geq 0} \tfrac{1}{2}\left\{-2\tilde{d}_i \tilde{u}_i' W \tilde{v}_i + \tilde{d}_i^2\right\} + \lambda \mathbf{p}(\tilde{d}_i).$$

The optimization problem analogous to (3.35) will be

$$\underset{\tilde{U},\tilde{V}}{\text{minimize}} \quad \tfrac{1}{2}\left\{\|W\|_F^2 - 2\sum_{i=1}^{n} \hat{d}_i \tilde{u}_i' W \tilde{v}_i + \sum_{i=1}^{n} \hat{d}_i^2\right\} + \lambda \sum_i \mathbf{p}(\hat{d}_i), \tag{3.37}$$

where $\hat{d}_i = S^{\mathbf{p}}_{\lambda}(\tilde{u}_i' W \tilde{v}_i)$, $\forall i$*. Any spectral function for which the above (3.37) is monotonically increasing in* $\tilde{u}_i' W \tilde{v}_i$ *for every* i *can be solved by a similar argument as given in the above proof. The solution will correspond to the first few largest left and right singular vectors of the matrix* W*. The optimal singular values will correspond to the relevant shrinkage/ threshold operator* $S^{\mathbf{p}}_{\lambda}(\cdot)$ *operated on the singular values of* W*. In particular for the indicator function* $\mathbf{p}(t) = \lambda\, \mathbf{1}(t \neq 0)$*, the top few singular values (un-shrunk) and the corresponding singular vectors is the solution.*

3.A.2 Proof of Lemma 8

This proof is based on sub-gradient characterizations and is inspired by some techniques used in Cai et al. (2010).

Proof. From Lemma 6, we know that if $\hat{Z}$ solves the problem (3.8), then it satisfies the sub-gradient stationary conditions:

$$0 \in -(W - \hat{Z}) + \lambda \partial \|\hat{Z}\|_*. \tag{3.38}$$

$\mathrm{S}_\lambda(W_1)$ and $\mathrm{S}_\lambda(W_2)$ solve the problem (3.8) with $W = W_1$ and $W = W_2$ respectively, hence (3.38) holds with $W = W_1$, $\hat{Z}_1 = \mathrm{S}_\lambda(W_1)$ and $W = W_2$, $\hat{Z}_2 = \mathrm{S}_\lambda(W_1)$.

The sub-gradients of the nuclear norm $\|Z\|_*$ are given by

$$\partial\|Z\|_* = \{UV' + \omega : \omega_{m\times n},\ U'\omega = 0,\ \omega V = 0,\ \|\omega\|_2 \leq 1\}, \tag{3.39}$$

where $Z = UDV'$ is the SVD of Z.

Let $p(\hat{Z}_i)$ denote an element in $\partial\|\hat{Z}_i\|_*$. Then

$$\hat{Z}_i - W_i + \lambda p(\hat{Z}_i) = 0,\ i = 1, 2.$$

The above gives

$$(\hat{Z}_1 - \hat{Z}_2) - (W_1 - W_2) + \lambda(p(\hat{Z}_1) - p(\hat{Z}_2)) = 0, \tag{3.40}$$

from which we obtain

$$\langle \hat{Z}_1 - \hat{Z}_2, \hat{Z}_1 - \hat{Z}_2\rangle - \langle W_1 - W_2, \hat{Z}_1 - \hat{Z}_2\rangle + \lambda\langle p(\hat{Z}_1) - p(\hat{Z}_2), \hat{Z}_1 - \hat{Z}_2\rangle = 0,$$

where $\langle a, b\rangle = \text{trace}(a'b)$.

Now observe that

$$\langle p(\hat{Z}_1) - p(\hat{Z}_2), \hat{Z}_1 - \hat{Z}_2\rangle = \langle p(\hat{Z}_1), \hat{Z}_1\rangle - \langle p(\hat{Z}_1), \hat{Z}_2\rangle - \langle p(\hat{Z}_2), \hat{Z}_1\rangle + \langle p(\hat{Z}_2), \hat{Z}_2\rangle. \tag{3.41}$$

By the characterization of subgradients as in (3.39), we have

$$\langle p(\hat{Z}_i), \hat{Z}_i\rangle = \|\hat{Z}_i\|_* \text{ and } \|p(\hat{Z}_i)\|_2 \leq 1,\ \ i = 1, 2.$$

This implies

$$|\langle p(\hat{Z}_i), \hat{Z}_j\rangle| \leq \|p(\hat{Z}_i)\|_2\|\hat{Z}_j\|_* \leq \|\hat{Z}_j\|_* \text{ for } i \neq j \in \{1, 2\}.$$

Using the above inequalities in (3.41) we obtain:

$$\langle p(\hat{Z}_1), \hat{Z}_1\rangle + \langle p(\hat{Z}_2), \hat{Z}_2\rangle = \|\hat{Z}_1\|_* + \|\hat{Z}_2\|_* \tag{3.42}$$

$$-\langle p(\hat{Z}_1), \hat{Z}_2\rangle - \langle p(\hat{Z}_2), \hat{Z}_1\rangle \geq -\|\hat{Z}_2\|_* - \|\hat{Z}_1\|_*. \tag{3.43}$$

Using (3.42,3.43) we see that the r.h.s. of (3.41) is non-negative. Hence

$$\langle p(\hat{Z}_1) - p(\hat{Z}_2), \hat{Z}_1 - \hat{Z}_2\rangle \geq 0.$$

Using the above in (3.40), we obtain:

$$\|\hat{Z}_1 - \hat{Z}_2\|_F^2 = \langle \hat{Z}_1 - \hat{Z}_2, \hat{Z}_1 - \hat{Z}_2\rangle \leq \langle W_1 - W_2, \hat{Z}_1 - \hat{Z}_2\rangle. \tag{3.44}$$

Using the Cauchy-Schwarz Inequality, $\|\hat{Z}_1 - \hat{Z}_2\|_F\|W_1 - W_2\|_F \geq \langle \hat{Z}_1 - \hat{Z}_2, W_1 - W_2\rangle$ in (3.44) we get

$$\|\hat{Z}_1 - \hat{Z}_2\|_F^2 \leq \langle \hat{Z}_1 - \hat{Z}_2, W_1 - W_2\rangle \leq \|\hat{Z}_1 - \hat{Z}_2\|_F\|W_1 - W_2\|_F$$

and in particular

$$\|\hat{Z}_1 - \hat{Z}_2\|_F^2 \leq \|\hat{Z}_1 - \hat{Z}_2\|_F\|W_1 - W_2\|_F.$$

The above further simplifies to

$$\|W_1 - W_2\|_F^2 \geq \|\hat{Z}_1 - \hat{Z}_2\|_F^2 = \|\mathrm{S}_\lambda(W_1) - \mathrm{S}_\lambda(W_2)\|_F^2.$$

$\square$

3.A.3 Proof of Lemma 9

Proof. We will first show (3.15) by observing the following inequalities:

$$\begin{aligned}
\|Z_\lambda^{k+1} - Z_\lambda^k\|_F^2 &= \|\mathrm{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k)) - \mathrm{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))\|_F^2 \\
\text{(by Lemma 8)} \quad &\leq \| \left(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k)\right) - \left(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1})\right) \|_F^2 \\
&= \|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^{k-1})\|_F^2 \qquad (3.45) \\
&\leq \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2. \qquad (3.46)
\end{aligned}$$

The above implies that the sequence $\{\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2\}$ converges (since it is decreasing and bounded below). We still require to show that $\{\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2\}$ converges to zero.

The convergence of $\{\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2\}$ implies that:

$$\|Z_\lambda^{k+1} - Z_\lambda^k\|_F^2 - \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2 \to 0 \text{ as } k \to \infty.$$

The above observation along with the inequality in (3.45,3.46) gives

$$\|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^{k-1})\|_F^2 - \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2 \to 0 \implies P_\Omega(Z_\lambda^k - Z_\lambda^{k-1}) \to 0, \qquad (3.47)$$

as $k \to \infty$.

Lemma 7 shows that the non-negative sequence $f_\lambda(Z_\lambda^k)$ is decreasing in k. So as $k \to \infty$ the sequence $f_\lambda(Z_\lambda^k)$ converges.

Furthermore from (3.13,3.14) we have

$$Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^k) - Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^{k+1}) \to 0 \text{ as } k \to \infty,$$

which implies that

$$\|P_\Omega^\perp(Z_\lambda^k) - P_\Omega^\perp(Z_\lambda^{k+1})\|_F^2 \to 0 \text{ as } k \to \infty.$$

The above along with (3.47) gives

$$Z_\lambda^k - Z_\lambda^{k-1} \to 0 \text{ as } k \to \infty.$$

This completes the proof. □

3.A.4 Proof of Lemma 10

Proof. The sub-gradients of the nuclear norm $\|Z\|_*$ are given by

$$\partial\|Z\|_* = \{UV' + W : W_{m\times n},\ U'W = 0,\ WV = 0,\ \|W\|_2 \leq 1\}, \tag{3.48}$$

where $Z = UDV'$ is the SVD of Z. Since Z_λ^k minimizes $Q_\lambda(Z|Z_\lambda^{k-1})$, it satisfies:

$$0 \in -(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}) - Z_\lambda^k) + \partial\|Z_\lambda^k\|_*\ \ \forall k. \tag{3.49}$$

Suppose Z^* is a limit point of the sequence Z_λ^k. Then there exists a subsequence $\{n_k\} \subset \{1, 2, \ldots\}$ such that $Z_\lambda^{n_k} \to Z^*$ as $k \to \infty$.

By Lemma 9 this subsequence $Z_\lambda^{n_k}$ satisfies

$$Z_\lambda^{n_k} - Z_\lambda^{n_k-1} \to 0$$

implying

$$P_\Omega^\perp(Z_\lambda^{n_k-1}) - Z_\lambda^{n_k} \longrightarrow P_\Omega^\perp(Z_\lambda^*) - Z_\lambda^* = -P_\Omega(Z^*).$$

Hence,

$$(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{n_k-1}) - Z_\lambda^{n_k}) \longrightarrow (P_\Omega(X) - P_\Omega(Z_\lambda^*)). \tag{3.50}$$

For every k, a sub-gradient $p(Z_\lambda^k) \in \partial\|Z_\lambda^k\|_*$ corresponds to a tuple (u_k, v_k, w_k) satisfying the properties of the set $\partial\|Z^k_\lambda\|_*$ (3.48).

Consider $p(Z^{n_k}_\lambda)$ along the sub-sequence n_k. As $n_k \longrightarrow \infty$, $Z^{n_k}_\lambda \longrightarrow Z^*_\lambda$.

Let

$$Z^{n_k}_\lambda = u_{n_k} D_{n_k} v'_{n_k}, \quad Z^* = u_\infty D^* v'_\infty$$

denote the SVD's. The product of the singular vectors converge $u'_{n_k} v_{n_k} \to u'_\infty v_\infty$. Furthermore due to boundedness (passing on to a further subsequence if necessary) $w_{n_k} \to w_\infty$. The limit $u_\infty v'_\infty + w_\infty$ clearly satisfies the criterion of being a sub-gradient of Z^*. Hence this limit corresponds to $p(Z^*_\lambda) \in \partial\|Z^*_\lambda\|_*$.

Furthermore from (3.49, 3.50), passing on to the limits along the subsequence n_k, we have

$$\mathbf{0} \in -(P_\Omega(X) - P_\Omega(Z^*_\lambda)) + \partial\|Z^*_\lambda\|_*.$$

Hence the limit point Z^*_λ is a stationary point of $f_\lambda(Z)$.

We shall now prove (3.16). We know that for every n_k

$$Z^{n_k}_\lambda = \mathrm{S}_\lambda(P_\Omega(X) + P^\perp_\Omega(Z^{n_k-1}_\lambda)). \tag{3.51}$$

From Lemma 9, we know $Z^{n_k}_\lambda - Z^{n_k-1}_\lambda \to 0$. This observation along with the continuity of $\mathrm{S}_\lambda(\cdot)$ gives

$$\mathrm{S}_\lambda(P_\Omega(X) + P^\perp_\Omega(Z^{n_k-1}_\lambda)) \to \mathrm{S}_\lambda(P_\Omega(X) + P^\perp_\Omega(Z^*_\lambda)).$$

Thus passing over to the limits on both sides of (3.51) we get

$$Z^*_\lambda = \mathrm{S}_\lambda(P_\Omega(X) + P^\perp_\Omega(Z^*_\lambda)),$$

therefore completing the proof. □

3.A.5 Proof of Lemma 11

The proof is motivated by the principle of embedding an arbitrary matrix into a positive semidefinite matrix (Fazel 2002). We require the following proposition, which we prove using techniques used in the same reference.

Proposition 1. *Suppose matrices* $W_{m\times m}, \tilde{W}_{n\times n}, Z_{m\times n}$ *satisfy the following:*

$$\begin{pmatrix} W & Z \\ Z^T & \tilde{W} \end{pmatrix} \succeq 0.$$

Then $\mathrm{trace}(W) + \mathrm{trace}(\tilde{W}) \geq 2\|Z\|_*$.

Proof. Let $Z_{m\times n} = L_{m\times r}\Sigma_{r\times r}R^T_{n\times r}$ denote the SVD of Z, where r is the rank of the matrix Z. Observe that the trace of the product of two positive semidefinite matrices is always non-negative. Hence we have the following inequality:

$$\mathrm{trace}\begin{pmatrix} LL^T & -LR^T \\ -RL^T & RR^T \end{pmatrix}\begin{pmatrix} W & Z \\ Z^T & \tilde{W} \end{pmatrix} \succeq 0.$$

Simplifying the above expression we get:

$$\mathrm{trace}(LL^TW) - \mathrm{trace}(LR^TZ^T) - \mathrm{trace}(RL^TZ) + \mathrm{trace}(RR^T\tilde{W}) \geq 0. \tag{3.52}$$

Due to the orthogonality of the columns of L, R we have the following inequalities:

$$\mathrm{trace}(LL^TW) \leq \mathrm{trace}(W) \;\text{ and }\; \mathrm{trace}(RR^T\tilde{W}) \leq \mathrm{trace}(\tilde{W}).$$

Furthermore, using the SVD of Z:

$$\mathrm{trace}(LR^TZ^T) = \mathrm{trace}(\Sigma) = \mathrm{trace}(LR^TZ^T).$$

Using the above in (3.52), we have:

$$\mathrm{trace}(W) + \mathrm{trace}(\tilde{W}) \geq 2\,\mathrm{trace}(\Sigma) = 2\,\|Z\|_*.$$

□

Proof of Lemma 11. For the matrix Z, consider any decomposition of the form $Z = \tilde{U}_{m\times r}\tilde{V}^T_{n\times r}$ and construct the following matrix

$$\begin{pmatrix} \tilde{U}\tilde{U}^T & Z \\ Z^T & \tilde{V}\tilde{V}^T \end{pmatrix} = \begin{pmatrix} \tilde{U} \\ \tilde{V} \end{pmatrix} (\tilde{U}^T\tilde{V}^T), \tag{3.53}$$

which is positive semidefinite. Applying Proposition 1 to the left hand matrix in (3.53), we have:

$$\mathrm{trace}(\tilde{U}\tilde{U}^T) + \mathrm{trace}(\tilde{V}\tilde{V}^T) \geq 2\,\|Z\|_*.$$

Minimizing both sides above w.r.t. the decompositions $Z = \tilde{U}_{m\times r}\tilde{V}^T_{n\times r}$; we have

$$\min_{\tilde{U},\tilde{V};\,Z=\tilde{U}\tilde{V}^T} \left\{\mathrm{trace}(\tilde{U}\tilde{U}^T) + \mathrm{trace}(\tilde{V}\tilde{V}^T)\right\} \geq 2\,\|Z\|_*. \tag{3.54}$$

Through the SVD of Z we now show that equality is attained in (3.54). Suppose Z is of rank $k \leq \min(m, n)$, and denote its SVD by $Z_{m\times n} = L_{m\times k}\Sigma_{k\times k}R^T_{n\times k}$. Then for $\tilde{U} = L_{m\times k}\Sigma^{\frac{1}{2}}_{k\times k}$ and $\tilde{V} = R_{n\times k}\Sigma^{\frac{1}{2}}_{k\times k}$ the equality in (3.54) is attained.

Hence, we have:

$$\begin{aligned} \|Z\|_* &= \min_{\tilde{U},\tilde{V};Z=\tilde{U}\tilde{V}^T} \left\{\mathrm{trace}(\tilde{U}\tilde{U}^T) + \mathrm{trace}(\tilde{V}\tilde{V}^T)\right\} \\ &= \min_{\tilde{U},\tilde{V};Z=\tilde{U}_{m\times k}\tilde{V}^T_{n\times k}} \left\{\mathrm{trace}(\tilde{U}\tilde{U}^T) + \mathrm{trace}(\tilde{V}\tilde{V}^T)\right\}. \end{aligned}$$

Note that the minimum can also be attained for matrices with $r \geq k$ or even $r \geq \min(m, n)$;

however, it suffices to consider matrices with $r = k$. Also it is easily seen that the minimum cannot be attained for any $r < k$; hence the minimal rank r for which (3.29) holds true is $r = k$. □

3.A.6 Proof of Theorem 7

There is a close resemblance between SOFT-IMPUTE and Nesterov's gradient method (Nesterov 2007, Section 3). However, as mentioned earlier the original motivation of our algorithm is very different.

The techniques used in this proof are adapted from Nesterov (2007).

Proof. Plugging $Z_\lambda^k = \tilde{Z}$ in (3.11), we have

$$\begin{aligned} Q_\lambda(Z|Z_\lambda^k) &= f_\lambda(Z_\lambda^k) + \tfrac{1}{2}\|P_\Omega^\perp(Z_\lambda^k - Z)\|_F^2 \\ &\geq f_\lambda(Z_\lambda^k). \end{aligned} \tag{3.55}$$

Let $Z_\lambda^k(\theta)$ denote a convex combination of the optimal solution (Z_λ^∞) and the k^{th} iterate (Z_λ^k):

$$Z_\lambda^k(\theta) = \theta Z_\lambda^\infty + (1-\theta) Z_\lambda^k. \tag{3.56}$$

Using the convexity of $f_\lambda(\cdot)$ we get:

$$f_\lambda(Z_\lambda^k(\theta)) \leq (1-\theta) f_\lambda(Z_\lambda^k) + \theta f_\lambda(Z_\lambda^\infty). \tag{3.57}$$

Expanding $Z_\lambda^k(\theta)$ using (3.56), and simplifying $P_\Omega^\perp(Z_\lambda^k - Z_\lambda^k(\theta))$ we have:

$$\begin{aligned} \|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^k(\theta))\|_F^2 &= \theta^2 \|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^\infty)\|_F^2 \\ &\leq \theta^2 \|Z_\lambda^k - Z_\lambda^\infty\|_F^2 && (3.58) \\ &\leq \theta^2 \|Z_\lambda^0 - Z_\lambda^\infty\|_F^2. && (3.59) \end{aligned}$$

Line 3.59 follows from (3.58) by observing that $\|Z_\lambda^m - Z_\lambda^\infty\|_F^2 \leq \|Z_\lambda^{m-1} - Z_\lambda^\infty\|_F^2$, $\forall m$—a consequence of the inequalities (3.19) and (3.18), established in Theorem 6.

Using (3.55), the value of $f_\lambda(Z)$ at the $(k+1)^{\text{th}}$ iterate satisfies the following chain of inequalities:

$$
\begin{aligned}
f_\lambda(Z_\lambda^{k+1}) &\leq \min_Z \left\{ f_\lambda(Z) + \tfrac{1}{2}\|P_\Omega^\perp(Z_\lambda^k - Z)\|_F^2 \right\} \\
&\leq \min_{\theta\in[0,1]} \left\{ f_\lambda(Z_\lambda^k(\theta)) + \tfrac{1}{2}\|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^k(\theta))\|_F^2 \right\} &\quad (3.60) \\
&\leq \min_{\theta\in[0,1]} \left\{ f_\lambda(Z_\lambda^k) + \theta(f_\lambda(Z_\lambda^\infty) - f_\lambda(Z_\lambda^k)) + \tfrac{1}{2}\theta^2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2 \right\}. &\quad (3.61)
\end{aligned}
$$

Line 3.61 follows from Line 3.60, by using (3.57) and (3.59).

The r.h.s. expression in (3.61), is minimized at $\widehat{\theta}(k+1)$ given by

$$
\begin{aligned}
\widehat{\theta}(k+1) &= \min\{1, \theta_k\} \in [0,1], \quad \text{where,} \\
\theta_k &= \frac{f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty)}{\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2}.
\end{aligned}
$$

If $\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2 = 0$, then we take $\theta_k = \infty$.

Note that θ_k is a decreasing sequence. This implies that if $\theta_{k_0} \leq 1$ then $\theta_m \leq 1$ for all $m \geq k_0$.

Suppose, $\theta_0 > 1$. Then $\widehat{\theta}(1) = 1$. Hence using (3.61) we have:

$$
f_\lambda(Z_\lambda^1) - f_\lambda(Z_\lambda^\infty) \leq \tfrac{1}{2}\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2 \implies \theta_1 \leq \tfrac{1}{2}.
$$

Thus we get back to the former case.

Hence $\theta_k \leq 1$ for all $k \geq 1$.

In addition, observe the previous deductions show that, if $\theta_0 > 1$ then (3.20) holds true for $k = 1$.

Combining the above observations, plugging in the value of $\widehat{\theta}$ in (3.61) and simplifying, we get:

$$f_\lambda(Z_\lambda^{k+1}) - f_\lambda(Z_\lambda^k) \leq -\frac{(f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty))^2}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2}. \tag{3.62}$$

For the sake of notational convenience, we define the sequence $\alpha_k = f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty)$. It is easily seen that α_k is a non-negative decreasing sequence.

Using this notation in (3.62) we get:

$$\begin{aligned} \alpha_k &\geq \frac{\alpha_k^2}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_{k+1} \\ (\text{Since } \alpha_k \downarrow) &\geq \frac{\alpha_k \alpha_{k+1}}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_{k+1}. \end{aligned} \tag{3.63}$$

Dividing both sides of the inequality in (3.63), by $\alpha_k \alpha_{k+1}$ we have:

$$\alpha_{k+1}^{-1} \geq \frac{1}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_k^{-1}. \tag{3.64}$$

Summing both sides of (3.64) over $1 \leq k \leq (k-1)$ we get:

$$\alpha_k^{-1} \geq \frac{k-1}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_1^{-1}. \tag{3.65}$$

Since $\theta_1 \leq 1$, we observe $\alpha_1/(2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2) \leq 1/2$—using this in (3.65) and rearranging terms we get, the desired inequality (3.20)—completing the proof of the Theorem. $\square$

Part III

Covariance Selection & Graphical Lasso : New Insights & Alternatives

This part of the thesis is dedicated to studying properties of the ℓ_1 penalized negative log-likelihood for learning undirected Gaussian graphical models, popularly dubbed as the Graphical Lasso (Friedman, Hastie & Tibshirani 2007, Banerjee et al. 2008) or GLASSO . It is divided into two chapters:

Chapter 4: *Exact* Covariance Thresholding for GLASSO :

This is about a previously unknown and interesting property of GLASSO which establishes the equivalence between the connected components of the sparse precision graph and the same of a thresholded version of the sample covariance matrix.

Chapter 5: GLASSO New Insights and Alternatives :

This chapter highlights some of the properties of the GLASSO algorithm that appear mysterious and providing explanations to them. In the process of our analysis, we propose new and superior algorithm(s) on the primal : P-GLASSO and DP-GLASSO and justify their usefulness.

Chapter 4

Exact Covariance Thresholding for GLASSO

Abstract: We consider the sparse inverse covariance regularization problem or *graphical lasso* with regularization parameter λ. Suppose the sample *covariance graph* formed by thresholding the entries of the sample covariance matrix at λ is decomposed into connected components. We show that the *vertex-partition* induced by the connected components of the thresholded sample covariance graph (at λ) is *exactly* equal to that induced by the connected components of the estimated concentration graph, obtained by solving the graphical lasso problem for the *same* λ. This characterizes a very interesting property of a path of graphical lasso solutions. Furthermore, this simple rule, when used as a wrapper around existing algorithms for the graphical lasso, leads to enormous performance gains. For a range of values of λ, our proposal splits a large graphical lasso problem into smaller tractable problems, making it possible to solve an otherwise infeasible large-scale problem. We illustrate the graceful scalability of our proposal via synthetic and real-life microarray examples.

This chapter is based on joint work with Trevor Hastie and appears in our paper Mazumder & Hastie (2012).

4.1 Introduction

Consider a data matrix $\mathbf{X}_{n\times p}$ comprising of n sample realizations from a p dimensional Gaussian distribution with zero mean and positive definite covariance matrix $\mathbf{\Sigma}_{p\times p}$ (unknown), i.e. $\mathbf{x}_i \stackrel{\text{i.i.d}}{\sim} \text{MVN}(\mathbf{0}, \mathbf{\Sigma}),\ i = 1, \ldots, n$. The task is to estimate the unknown $\mathbf{\Sigma}$ based on the n samples. ℓ_1 regularized Sparse Inverse Covariance Selection also known as *graphical lasso* (Friedman, Hastie & Tibshirani 2007, Banerjee et al. 2008, Yuan & Lin 2007) estimates the covariance matrix $\mathbf{\Sigma}$, under the assumption that the inverse covariance matrix i.e. $\mathbf{\Sigma^{-1}}$ is sparse. This is achieved by minimizing the regularized negative log-likelihood function:

$$\underset{\mathbf{\Theta}\succeq\mathbf{0}}{\text{minimize}} \quad -\log\det(\mathbf{\Theta}) + \text{ tr}(\text{S}\mathbf{\Theta}) + \lambda\sum_{i,j}|\theta_{ij}|, \tag{4.1}$$

where $\mathbf{S}$ is the sample covariance matrix and θ_{ij} is the (i,j)th entry of the matrix $\mathbf{\Theta}$. Problem (4.1) is a convex optimization problem in the variable $\mathbf{\Theta}$ (Boyd & Vandenberghe 2004). Let $\widehat{\mathbf{\Theta}}^{(\lambda)}$ denote the solution to (4.1). We note that (4.1) can also be used in a more non-parametric fashion for any positive semi-definite input matrix S, not necessarily a sample covariance matrix of a MVN sample as described above.

A related criterion to (4.1) is one where the diagonals are not penalized—by substituting $\text{S} \leftarrow \text{S} + \lambda I_{p\times p}$ in the "unpenalized" problem we get (4.1). In this chapter we concentrate on problem (4.1).

Developing efficient large-scale algorithms for (4.1) is an active area of research across the fields of Convex Optimization, Machine Learning and Statistics. Many algorithms have been proposed for this task (Friedman, Hastie & Tibshirani 2007, Banerjee et al. 2008, Lu 2009, Lu 2010, Scheinberg et al. 2010, Yuan 2009, for example). However, it appears that certain special properties of the solution to (4.1) have been largely ignored. This chapter is about one such (surprising) property—namely establishing an equivalence between the *vertex-partition* induced by the connected components of the non-zero pattern of $\widehat{\mathbf{\Theta}}^{(\lambda)}$ and the thresholded sample covariance matrix S. This chapter is *not* about a specific algorithm for the problem (4.1)—it focuses on the aforementioned observation that leads to a novel thresholding/screening procedure based on S. This provides interesting insight into the path of solutions $\{\widehat{\mathbf{\Theta}}^{(\lambda)}\}_{\lambda \geq 0}$ obtained by solving (4.1), over a path of λ values. The behavior of the connected-components obtained from the non-zero patterns of $\{\widehat{\mathbf{\Theta}}^{(\lambda)}\}_{\lambda \geq 0}$ can be completely understood by simple screening rules on S. This can be done without *even attempting* to solve (4.1)—arguably a very challenging convex optimization problem. Furthermore, this thresholding rule can be used as a *wrapper* to enormously boost the performance of existing algorithms, as seen in our experiments. This strategy becomes extremely effective in solving large problems over a range of values of λ—sufficiently restricted to ensure sparsity and the separation into connected components. Of course, for sufficiently small values of λ there will be no separation into components, and hence no computational savings.

At this point we introduce some notation and terminology, which we will use throughout the chapter.

4.1.1 Notations and preliminaries

For a matrix $\mathbf{Z}$, its $(i,j)^{\text{th}}$ entry is denoted by z_{ij}.

We also introduce some graph theory notations and definitions (Bollobas 1998,) sufficient for this exposition. A finite undirected graph $\mathcal{G}$ on p vertices is given by the ordered tuple $\mathcal{G} = (\mathcal{V}, \boldsymbol{\mathcal{E}})$, where $\mathcal{V}$ is the set of nodes and $\boldsymbol{\mathcal{E}}$ the collection of (undirected) edges. The edge-set is equivalently represented via a (symmetric) 0-1 matrix[1] (also known as the *adjacency* matrix) with p rows/columns. We use the convention that a node is not connected to itself, so the diagonals of the adjacency matrix are all zeros. Let $|\mathcal{V}|$ and $|\boldsymbol{\mathcal{E}}|$ denote the number of nodes and edges respectively.

We say two nodes $u, v \in \mathcal{V}$ are *connected* if there is a *path* between them. A maximal

[1] 0 denotes absence of an edge and 1 denotes its presence.

connected *subgraph*[2] is a *connected component* of the graph $\mathcal{G}$. *Connectedness* is an equivalence relation that decomposes a graph $\mathcal{G}$ into its connected components $\{(\mathcal{V}_\ell, \boldsymbol{\mathcal{E}}_\ell)\}_{1\leq\ell\leq K}$—with $\mathcal{G} = \cup_{\ell=1}^{K}(\mathcal{V}_\ell, \boldsymbol{\mathcal{E}}_\ell)$, where K denotes the number of connected components. This decomposition partitions the vertices $\mathcal{V}$ of $\mathcal{G}$ into $\{\mathcal{V}_\ell\}_{1\leq\ell\leq K}$. Note that the labeling of the components is unique upto permutations on $\{1,\ldots,K\}$. Throughout this chapter we will often refer to this partition as the *vertex-partition* induced by the components of the graph $\mathcal{G}$. If the size of a component is one i.e. $|\mathcal{V}_\ell| = 1$, we say that the node is *isolated*. Suppose a graph $\widehat{\mathcal{G}}$ defined on the set of vertices $\mathcal{V}$ admits the following decomposition into connected components: $\widehat{\mathcal{G}} = \cup_{\ell=1}^{\widehat{K}}(\widehat{\mathcal{V}}_\ell, \widehat{\boldsymbol{\mathcal{E}}}_\ell)$. We say the vertex-partitions induced by the connected components of $\mathcal{G}$ and $\widehat{\mathcal{G}}$ are *equal* if $\widehat{K} = K$ and there is a permutation π on $\{1,\ldots,K\}$ such that $\widehat{\mathcal{V}}_{\pi(\ell)} = \mathcal{V}_\ell$ for all $\ell \in \{1,\ldots,K\}$.

The chapter is organized as follows. Section 4.2 describes the covariance graph thresholding idea along with theoretical justification and related work, followed by complexity analysis of the algorithmic framework in Section 4.3. Numerical experiments appear in Section 4.4, concluding remarks in Section 4.5 and the proofs are gathered in the Appendix 4.A.

4.2 Methodology: *Exact* Thresholding of the Covariance Graph

The sparsity pattern of the solution $\widehat{\boldsymbol{\Theta}}^{(\lambda)}$ to (4.1) gives rise to the symmetric edge matrix/skeleton $\in \{0,1\}^{p\times p}$ defined by:

$$\boldsymbol{\mathcal{E}}_{ij}^{(\lambda)} = \begin{cases} 1 & \text{if } \widehat{\theta}_{ij}^{(\lambda)} \neq 0,\ i \neq j; \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The above defines a symmetric graph $\mathcal{G}^{(\lambda)} = (\mathcal{V}, \boldsymbol{\mathcal{E}}^{(\lambda)})$, namely the *estimated concentration graph* (Cox & Wermuth 1996, Lauritzen 1996) defined on the nodes $\mathcal{V} = \{1,\ldots,p\}$ with

[2] $\mathcal{G}' = (\mathcal{V}', \boldsymbol{\mathcal{E}}')$ is a *subgraph* of $\mathcal{G}$ if $\mathcal{V}' \subset \mathcal{V}$ and $\boldsymbol{\mathcal{E}}' \subset \boldsymbol{\mathcal{E}}$.

edges $\boldsymbol{\mathcal{E}}^{(\lambda)}$.

Suppose the graph $\mathcal{G}^{(\lambda)}$ admits a decomposition into $\kappa(\lambda)$ connected components:

$$\mathcal{G}^{(\lambda)} = \cup_{\ell=1}^{\kappa(\lambda)} \mathcal{G}_\ell^{(\lambda)}, \tag{4.3}$$

where $\mathcal{G}_\ell^{(\lambda)} = (\widehat{\mathcal{V}}_\ell^{(\lambda)}, \boldsymbol{\mathcal{E}}_\ell^{(\lambda)})$ are the components of the graph $\mathcal{G}^{(\lambda)}$. Note that $\kappa(\lambda) \in \{1, \ldots, p\}$, with $\kappa(\lambda) = p$ (large λ) implying that all nodes are isolated and for small enough values of λ, there is only one component i.e. $\kappa(\lambda) = 1$.

We now describe the simple screening/thresholding rule. Given λ, we perform a thresholding on the entries of the sample covariance matrix S and obtain a graph edge skeleton $\mathbf{E}^{(\lambda)} \in \{0, 1\}^{p \times p}$ defined by:

$$\mathbf{E}_{ij}^{(\lambda)} = \begin{cases} 1 & \text{if } |s_{ij}| > \lambda,\ i \neq j; \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

The symmetric matrix $\mathbf{E}^{(\lambda)}$ defines a symmetric graph on the nodes $\mathcal{V} = \{1, \ldots, p\}$ given by $\mathrm{G}^{(\lambda)} = (\mathcal{V}, \mathbf{E}^{(\lambda)})$. We refer to this as the *thresholded sample covariance graph*. Similar to the decomposition in (4.3), the graph $\mathrm{G}^{(\lambda)}$ also admits a decomposition into connected components:

$$\mathrm{G}^{(\lambda)} = \cup_{\ell=1}^{k(\lambda)} \mathrm{G}_\ell^{(\lambda)}, \tag{4.5}$$

where $\mathrm{G}_\ell^{(\lambda)} = (\mathcal{V}_\ell^{(\lambda)}, \mathbf{E}_\ell^{(\lambda)})$ are the components of the graph $\mathrm{G}^{(\lambda)}$.

Note that the components of $\mathcal{G}^{(\lambda)}$ require knowledge of $\widehat{\boldsymbol{\Theta}}^{(\lambda)}$—the solution to (4.1). Construction of $\mathrm{G}^{(\lambda)}$ and its components require operating on S—an operation that can be performed completely independent of the optimization problem (4.1), which is arguably more expensive (See Section 4.3). The surprising message we describe in this chapter is that the *vertex-partition* of the connected components of (4.5) is *exactly* equal to that of (4.3).

This observation has the following consequences:

1. We obtain a very interesting property of the path of solutions $\{\widehat{\boldsymbol{\Theta}}^{(\lambda)}\}_{\lambda \geq 0}$—the behavior of the connected components of the estimated concentration graph can be completely understood by simple screening rules on S.

2. The cost of computing the connected components of the thresholded sample covariance graph (4.5) is orders of magnitude smaller than the cost of fitting graphical models (4.1). Furthermore, the computations pertaining to the covariance graph can be done off-line and are amenable to parallel computation (See Section 4.3).

3. The optimization problem (4.1) completely separates into $k(\lambda)$ separate optimization sub-problems of the form (4.1). The sub-problems have size equal to the number of nodes in each component $p_i := |\mathcal{V}_i|, i = 1, \ldots, k(\lambda)$. Hence for certain values of λ, solving problem (4.1) becomes feasible although it may be impossible to operate on the $p \times p$ dimensional (global) variable $\boldsymbol{\Theta}$ on a single machine.

4. Suppose that for λ_0, there are $k(\lambda_0)$ components and the graphical model computations are distributed[3]. Since the vertex-partitions induced via (4.3) and (4.5) are nested with increasing λ (see Theorem 10), it suffices to operate independently on these separate machines to obtain the path of solutions $\{\widehat{\boldsymbol{\Theta}}^{(\lambda)}\}_{\lambda}$ for all $\lambda \geq \lambda_0$.

5. Consider a distributed computing architecture, where every machine allows operating on a graphical lasso problem (4.1) of maximal size $p_{\max}$. Then with relatively small effort we can find the smallest value of $\lambda = \lambda_{p_{\max}}$, such that there are no connected components of size larger than $p_{\max}$. Problem (4.1) thus 'splits up' independently into manageable problems across the different machines. When this structure is not exploited the global problem (4.1) remains intractable.

[3]Distributing these operations depend upon the number of processors available, their capacities, communication lag, the number of components and the maximal size of the blocks across all machines. These of-course depend upon the computing environment. In the context of the present problem, it is often desirable to club smaller components into a single machine.

The following theorem establishes the main technical contribution of this chapter—the equivalence of the vertex-partitions induced by the connected components of the thresholded sample covariance graph and the estimated concentration graph.

Theorem 9. *For any $\lambda > 0$, the components of the estimated concentration graph $\mathcal{G}^{(\lambda)}$, as defined in (4.2) and (4.3) induce* exactly *the same vertex-partition as that of the thresholded sample covariance graph $\mathrm{G}^{(\lambda)}$, defined in (4.4) and (4.5). That is $\kappa(\lambda) = k(\lambda)$ and there exists a permutation π on $\{1, \ldots, k(\lambda)\}$ such that:*

$$\widehat{\mathcal{V}}_i^{(\lambda)} = \mathcal{V}_{\pi(i)}^{(\lambda)}, \;\; \forall i = 1, \ldots, k(\lambda). \tag{4.6}$$

Proof. The proof of the theorem appears in Appendix 4.A.1. □

Since the decomposition of a symmetric graph into its connected components depends upon the ordering/ labeling of the components, the permutation π appears in Theorem 9.

Remark 2. *Note that the edge-structures* within *each block need not be preserved. Under a matching reordering of the labels of the components of $\mathcal{G}^{(\lambda)}$ and $\mathrm{G}^{(\lambda)}$:*
for every fixed ℓ such that $\widehat{\mathcal{V}}_\ell^{(\lambda)} = \mathcal{V}_\ell^{(\lambda)}$ the edge-sets $\boldsymbol{\mathcal{E}}_\ell^{(\lambda)}$ and $\mathbf{E}_\ell^{(\lambda)}$ are not *necessarily equal.*

Theorem 9 leads to a special property of the path-of-solutions to (4.1), i.e. the vertex-partition induced by the connected components of $\mathcal{G}^{(\lambda)}$ are nested with increasing λ. This is the content of the following theorem.

Theorem 10. *Consider two values of the regularization parameter such that $\lambda > \lambda' > 0$, with corresponding concentration graphs $\mathcal{G}^{(\lambda)}$ and $\mathcal{G}^{(\lambda')}$ as in (4.2) and connected components (4.3). Then the vertex-partition induced by the components of $\mathcal{G}^{(\lambda)}$ are* nested *within the partition induced by the components of $\mathcal{G}^{(\lambda')}$. Formally, $\kappa(\lambda) \geq \kappa(\lambda')$ and the vertex-partition $\{\widehat{\mathcal{V}}_\ell^{(\lambda)}\}_{1 \leq \ell \leq \kappa(\lambda)}$ forms a finer resolution of $\{\widehat{\mathcal{V}}_\ell^{(\lambda')}\}_{1 \leq \ell \leq \kappa(\lambda')}$.*

Proof. The proof of this theorem appears in the Appendix 4.A.2. □

Remark 3. *It is worth noting that Theorem 10 addresses the nesting of the edges* across *connected components and not within a component. In general, the edge-set $\boldsymbol{\mathcal{E}}^{(\lambda)}$ of the estimated concentration graph need not be nested as a function of λ:*
for $\lambda > \lambda'$, in general, $\boldsymbol{\mathcal{E}}^{(\lambda)} \not\subset \boldsymbol{\mathcal{E}}^{(\lambda')}$.

See Friedman, Hastie & Tibshirani (2007, Figure 3), for numerical examples demonstrating the non-monotonicity of the edge-set across λ, as described in Remark 3.

Node-Thresholding

A simple consequence of Theorem 9 is that of *node-thresholding*. If $\lambda \geq \max_{j \neq i} |\mathrm{S}_{ij}|$, then the i^{th} node will be isolated from the other nodes, the off-diagonal entries of the i^{th} row/column are all zero i.e. $\max_{j \neq i} |\widehat{\theta}^{(\lambda)}_{ij}| = 0$. Furthermore, the i^{th} diagonal entries of the estimated covariance and precision matrices are given by $(s_{ii} + \lambda)$ and $\frac{1}{s_{ii}+\lambda}$, respectively. Hence, as soon as $\lambda \geq \max_{i=1,\ldots,p}\{\max_{j \neq i} |s_{ij}|\}$, the estimated covariance and precision matrices obtained from (4.1) are both diagonal.

4.2.1 Related Work

Witten et al. (2011) independently discovered block screening as described in this chapter. At the time of our writing, an earlier version of their chapter was available (Witten & Friedman 2011); it proposed a scheme to detect isolated nodes for problem (4.1) via a simple screening of the entries of $\mathbf{S}$, but no block screening. Earlier, Banerjee et al. (2008, Theorem 4) made the same observation about isolated nodes. The revised manuscript (Witten et al. 2011) that includes block screening became available shortly after our chapter was submitted for publication.

Zhou et al. (2011) use a thresholding strategy followed by re-fitting for estimating Gaussian graphical models. Their approach is based on the node-wise lasso-regression procedure of Meinshausen & Bühlmann (2006). A hard thresholding is performed on the ℓ_1-penalized regression coefficient estimates at every node to obtain the graph structure. A restricted MLE for the concentration matrix is obtained for the graph. The proposal in our chapter differs since we are interested in solving the GLASSO problem (4.1).

4.3 Computational Complexity

The overall complexity of our proposal depends upon (a) the graph partition stage and (b) solving (sub)problems of the form (4.1). In addition to these, there is an unavoidable complexity associated with handling and/or forming S.

The cost of computing the connected components of the thresholded covariance graph is fairly negligible when compared to solving a similar sized graphical lasso problem (4.1)—see also our simulation studies in Section 4.4. In case we observe samples $\mathbf{x}_i \in \Re^p, i = 1, \ldots, n$ the cost for creating the sample covariance matrix S is $O(n \cdot p^2)$. Thresholding the sample covariance matrix costs $O(p^2)$. Obtaining the connected components of the thresholded covariance graph costs $O(|\mathbf{E}^{(\lambda)}| + p)$ (Tarjan 1972). Since we are interested in a region where the thresholded covariance graph is sparse enough to be broken into smaller connected components—$|\mathbf{E}^{(\lambda)}| \ll p^2$. Note that all computations pertaining to the construction of the connected components and the task of computing S can be computed off-line. Furthermore the computations are parallelizable. Gazit (1991, for example) describes parallel algorithms for computing connected components of a graph—they have a time complexity $O(\log(p))$ and require $O((|\mathbf{E}^{(\lambda)}| + p)/\log(p))$ processors with space $O(p + |\mathbf{E}^{(\lambda)}|)$.

There are a wide variety of algorithms for the task of solving (4.1). While an exhaustive review of the computational complexities of the different algorithms is beyond the scope of this chapter, we provide a brief summary for a few algorithms below.

Banerjee et al. (2008) proposed a smooth accelerated gradient based method (Nesterov 2005) with complexity $O(\frac{p^{4.5}}{\epsilon})$ to obtain an ϵ accurate solution—the per iteration cost being $O(p^3)$. They also proposed a block coordinate method which has a complexity of $O(p^4)$.

The complexity of the GLASSO algorithm (Friedman, Hastie & Tibshirani 2007) which uses a row-by-row block coordinate method is roughly $O(p^3)$ for reasonably sparse-problems with p nodes. For denser problems the cost can be as large as $O(p^4)$.

The algorithm SMACS proposed in Lu (2010) has a per iteration complexity of $O(p^3)$ and an overall complexity of $O(\frac{p^4}{\sqrt{\epsilon}})$ to obtain an $\epsilon > 0$ accurate solution.

It appears that most existing algorithms for (4.1), have a complexity of at least $O(p^3)$ to $O(p^4)$ or possibly larger, depending upon the algorithm used and the desired accuracy of the solution—making computations for (4.1) almost impractical for values of p much larger than 2000.

It is quite clear that the role played by covariance thresholding is indeed crucial in this context. Assume that we use a solver of complexity $O(p^J)$ with $J \in \{3, 4\}$, along with our screening procedure. Suppose for a given λ, the thresholded sample covariance graph has $k(\lambda)$ components—the total cost of solving these smaller problems is then $\sum_{i=1}^{k(\lambda)} O(|\mathcal{V}_i^{(\lambda)}|^J) \ll O(p^J)$, with $J \in \{3, 4\}$. This difference in practice can be enormous—see Section 4.4 for numerical examples. This is what makes large scale graphical lasso problems solvable!

4.4 Numerical examples

In this section we show via numerical experiments that the screening property helps in obtaining many fold speed-ups when compared to an algorithm that does not exploit it. Section 4.4.1 considers synthetic examples and Section 4.4.2 discusses real-life microarray data-examples.

4.4.1 Synthetic examples

Experiments are performed with two publicly available algorithm implementations for the problem (4.1):

GLASSO**:** The algorithm of Friedman, Hastie & Tibshirani (2007). We used the MATLAB wrapper available at `http://www-stat.stanford.edu/~tibs/glasso/index.html` to the Fortran code. The specific criterion for convergence (lack of progress of the diagonal entries) was set to 10^{-5} and the maximal number of iterations was set to 1000.

SMACS**:** denotes the algorithm of Lu (2010). We used the MATLAB implementation `smooth_covsel` available at `http://people.math.sfu.ca/~zhaosong/Codes/SMOOTH_COVSEL/`. The criterion for convergence (based on duality gap) was set to 10^{-5} and the maximal number of iterations was set to 1000.

We will like to note that the convergence criteria of the two algorithms GLASSO and SMACS are not the same. For obtaining the connected components of a symmetric adjacency matrix we used the MATLAB function `graphconncomp`. All of our computations are done in MATLAB 7.11.0 on a 3.3 GhZ Intel Xeon processor.

The simulation examples are created as follows. We generated a block diagonal matrix given by $\tilde{\mathrm{S}} = \mathrm{blkdiag}(\tilde{\mathrm{S}}_1, \ldots, \tilde{\mathrm{S}}_K)$, where each block $\tilde{\mathrm{S}}_\ell = \mathbf{1}_{p_\ell \times p_\ell}$—a matrix of all ones and $\sum_\ell p_\ell = p$. In the examples we took all p_ℓs to be equal to p_1 (say). Noise of the form $\sigma \cdot UU'$ (U is a $p \times p$ matrix with i.i.d. standard Gaussian entries) is added to $\tilde{\mathrm{S}}$ such that 1.25 times the largest (in absolute value) off block-diagonal (as in the block structure of $\tilde{\mathrm{S}}$) entry of $\sigma \cdot UU'$ equals the smallest absolute non-zero entry in $\tilde{\mathrm{S}}$ i.e. one. The sample covariance matrix is $\mathrm{S} = \tilde{\mathrm{S}} + \sigma \cdot UU'$.

We consider a number of examples for varying K and p_1 values, as shown in Table 4.1. Sizes were chosen such that it is at-least 'conceivable' to solve (4.1) on the full dimensional problem, without screening. In all the examples shown in Table 4.1, we set $\lambda_I := (\lambda_{\max} + \lambda_{\min})/2$, where for all values of λ in the interval $[\lambda_{\min}, \lambda_{\max}]$ the thresh-holded version of the sample covariance matrix has exactly K connected components. We also took a larger value of λ i.e. $\lambda_{II} := \lambda_{\max}$, which gave sparser estimates of the precision matrix but the number of connected components were the same.

The computations across different connected blocks could be distributed into as many machines. This would lead to almost a K fold improvement in timings, however in Table 4.1 we report the timings by operating serially across the blocks. The serial 'loop' across the different blocks are implemented in MATLAB.

Table 4.1 shows the rather remarkable improvements obtained by using our proposed covariance thresholding strategy as compared to operating on the whole matrix. Timing comparisons between GLASSO and SMACS are not fair, since GLASSO is written in Fortran and SMACS in MATLAB. However, we note that our experiments are meant to demonstrate how the thresholding helps in improving the overall computational time over the baseline method of not exploiting screening. Clearly our proposed strategy makes solving larger problems (4.1), not only feasible but with quite attractive computational time. The time taken by the graph-partitioning step in splitting the thresholded covariance graph into its connected components is negligible as compared to the timings for the optimization problem.

4.4.2 Micro-array Data Examples

The graphical lasso is often used in learning connectivity networks in gene-microarray data (Friedman, Hastie & Tibshirani 2007, see for example). Since in most real examples the number of genes p is around tens of thousands, obtaining an inverse covariance matrix by solving (4.1) is computationally impractical. The covariance thresholding method we propose easily applies to these problems—and as we see gracefully delivers solutions over a large range of the parameter λ. We study three different micro-array examples and observe that as one varies λ from large to small values, the thresholded covariance graph splits into a number of non-trivial connected components of varying sizes. We continue till a small/moderate value of λ when the maximal size of a connected component gets larger than a predefined machine-capacity or the 'computational budget' for a single graphical lasso problem. Note that in relevant micro-array applications, since $p \gg n$ (n, the number

K	p_1 / p	λ	**Algorithm**	**Algorithm Timings** (sec) with screen	without screen	**Ratio** Speedup factor	**Time** (sec) graph partition
2	200 / 400	λ_I	GLASSO	11.1	25.97	2.33	0.04
			SMACS	12.31	137.45	11.16	
		λ_{II}	GLASSO	1.687	4.783	2.83	0.066
			SMACS	10.01	42.08	4.20	
2	500 /1000	λ_I	GLASSO	305.24	735.39	2.40	0.247
			SMACS	175	2138*	12.21	
		λ_{II}	GLASSO	29.8	121.8	4.08	0.35
			SMACS	272.6	1247.1	4.57	
5	300 /1500	λ_I	GLASSO	210.86	1439	6.82	0.18
			SMACS	63.22	6062*	95.88	
		λ_{II}	GLASSO	10.47	293.63	28.04	0.123
			SMACS	219.72	6061.6	27.58	
5	500 /2500	λ_I	GLASSO	1386.9	-	-	0.71
			SMACS	493	-	-	
		λ_{II}	GLASSO	17.79	963.92	54.18	0.018
			SMACS	354.81	-	-	
8	300 /2400	λ_I	GLASSO	692.25	-	-	0.713
			SMACS	185.75	-	-	
		λ_{II}	GLASSO	9.07	842.7	92.91	0.023
			SMACS	153.55	-	-	

Table 4.1: Table showing (a) the times in seconds with screening, (b) without screening i.e. on the whole matrix and (c) the ratio (b)/(a) – 'Speedup factor' for algorithms GLASSO and SMACS. Algorithms with screening are operated serially—the times reflect the total time summed across all blocks. The column 'graph partition' lists the time for computing the connected components of the thresholded sample covariance graph. Since $\lambda_{II} > \lambda_I$, the former gives sparser models. '*' denotes the algorithm did not converge within 1000 iterations. '-' refers to cases where the respective algorithms failed to converge within 2 hours.

of samples is at most a few hundred) heavy regularization is required to control the variance of the covariance estimates—so it does seem reasonable to restrict to solutions of (4.1) for large values of λ.

Following are the data-sets we used for our experiments:

(A) This data-set appears in Alon et al. (1999) and has been analyzed by Rothman et al. (2008, for example). In this experiment, tissue samples were analyzed using an Affymetrix Oligonucleotide array. The data were processed, filtered and reduced to a subset of $p = 2000$ gene expression values. The number of Colon Adenocarcinoma tissue samples is $n = 62$.

(B) This is an early example of an expression array, obtained from the Patrick Brown Laboratory at Stanford University. There are $n = 385$ patient samples of tissue from various regions of the body (some from tumors, some not), with gene-expression measurements for $p = 4718$ genes.

(C) The third example is the by now famous NKI dataset that produced the 70-gene prognostic signature for breast cancer van de Vijver et al. (2002). Here there are $n = 295$ samples and $p = 24481$ genes.

Among the above, both (B) and (C) have few missing values—which we imputed by the respective global means of the observed expression values. For each of the three data-sets, we took S to be the corresponding sample correlation matrix. The *exact thresholding* methodolgy could have also been applied to the sample covariance matrix. Since it is a common practice to standardize the "genes", we operate on the sample correlation matrix.

Figure 4.1 shows how the component sizes of the thresholded covariance graph change across λ. We describe the strategy we used to arrive at the figure. Note that the connected components change *only* at the absolute values of the entries of S. From the sorted absolute values of the off-diagonal entries of S, we obtained the smallest value of λ, say $\lambda'_{\min}$, for which the size of the maximal connected component was 1500. For a grid of values of λ till $\lambda'_{\min}$, we computed the connected components of the thresholded sample-covariance matrix and obtained the size-distribution of the various connected components. Figure 4.1 shows how these components change over a range of values of λ for the three examples (A), (B) and (C). The number of connected components of a particular size is denoted by a color-scheme, described by the color-bar in the figures. With increasing λ: the larger connected components gradually disappear as they decompose into smaller components; the sizes of the connected components decrease and the frequency of the smaller components increase. Since these are all correlation matrices, for $\lambda \geq 1$ all the nodes in the graph become isolated. The range of λ values for which the maximal size of the components is smaller than 1500 differ across the three examples. For (C) there is a greater variety in the sizes of the components as compared to (A) and (B). Note that by Theorem 9, the pattern of the components appearing in Figure 4.1 are exactly the same as the components appearing in the solution of (4.1) for that λ.

For examples (B) and (C) we found that the full problem sizes are beyond the scope of GLASSO and SMACS—the screening rule is apparently the *only* way to obtain solutions for a reasonable range of λ-values as shown in Figure 4.1.

4.5 Conclusions

In this chapter we present a novel property characterizing the family of solutions to the graphical lasso problem (4.1), as a function of the regularization parameter λ. The property is fairly surprising—the vertex partition induced by the connected components of the non-zero pattern of the estimated concentration matrix (at λ) and the thresholded sample covariance matrix S (at λ) are *exactly equal*. This property seems to have been unobserved in the literature. Our observation not only provides interesting insights into the properties of the graphical lasso solution-path but also opens the door to solving large-scale graphical

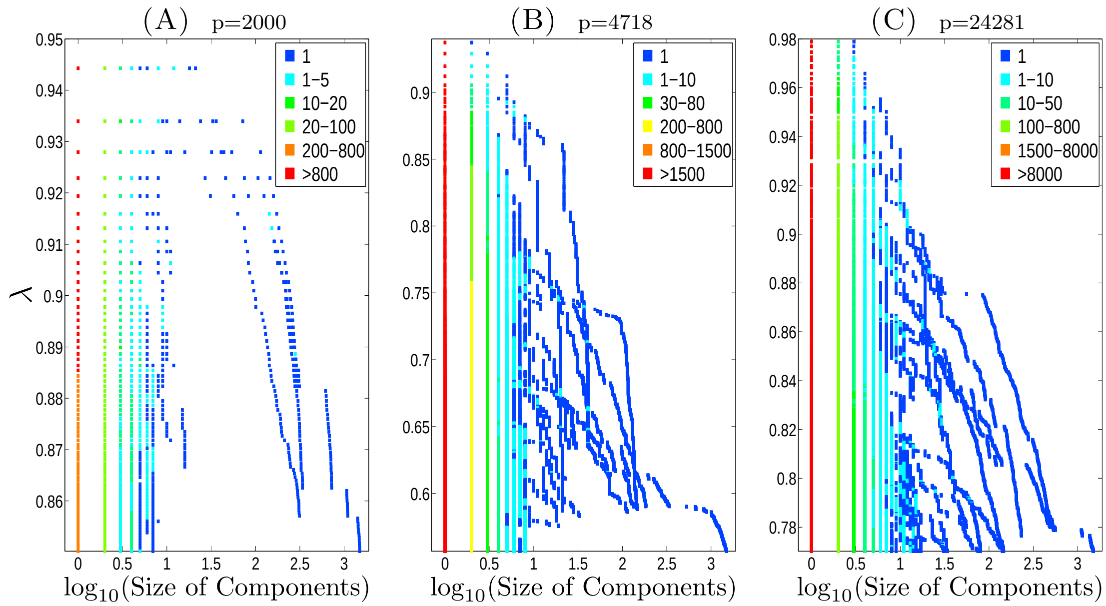

Figure 4.1: Figure showing the size distribution (in the log-scale) of connected components arising from the thresholded sample covariance graph for examples (A)-(C). For every value of λ (vertical axis), the horizontal slice denotes the sizes of the different components appearing in the thresholded covariance graph. The colors represent the number of components in the graph having that specific size. For every figure, the range of λ values is chosen such that the maximal size of the connected components do not exceed 1500.

lasso problems, which are otherwise intractable. This simple rule when used as a wrapper around existing algorithms leads to enormous performance boosts—on occasions by a factor of thousands!

4.A Appendix to Chapter 4

4.A.1 Proof of Theorem 9

Proof. Suppose $\widehat{\mathbf{\Theta}}$ (we suppress the superscript λ for notational convenience) solves problem (4.1), then standard KKT conditions of optimality (Boyd & Vandenberghe 2004) give:

$$|s_{ij} - \widehat{w}_{ij}| \leq \lambda \quad \forall\, \widehat{\theta}_{ij} = 0; \qquad \text{and} \tag{4.7}$$

$$\widehat{w}_{ij} = s_{ij} + \lambda \quad \forall\, \widehat{\theta}_{ij} > 0; \quad \widehat{w}_{ij} = s_{ij} - \lambda \,\forall\, \widehat{\theta}_{ij} < 0; \tag{4.8}$$

where $\widehat{\mathbf{W}} = (\widehat{\boldsymbol{\Theta}})^{-1}$. The diagonal entries satisfy $\widehat{w}_{ii} = s_{ii} + \lambda$, for $i = 1, \ldots, p$.

Using (4.4) and (4.5), there exists an ordering of the vertices $\{1, \ldots, p\}$ of the graph such that $\mathbf{E}^{(\lambda)}$ is block-diagonal. For notational convenience, we will assume that the matrix is already in that order. Under this ordering of the vertices, the edge-matrix of the thresholded covariance graph is of the form:

$$\mathbf{E}^{(\lambda)} = \begin{pmatrix} \mathbf{E}_1^{(\lambda)} & 0 & \cdots & 0 \\ 0 & \mathbf{E}_2^{(\lambda)} & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{E}_{k(\lambda)}^{(\lambda)} \end{pmatrix} \tag{4.9}$$

where the different components represent blocks of indices given by: $\mathcal{V}_\ell^{(\lambda)}, \ell = 1, \ldots, k(\lambda)$.

We will construct a matrix $\widehat{\mathbf{W}}$ having the same structure as (4.9) which is a solution to (4.1). Note that if $\widehat{\mathbf{W}}$ is block diagonal then so is its inverse. Let $\widehat{\mathbf{W}}$ and its inverse $\widehat{\boldsymbol{\Theta}}$ be given by:

$$\widehat{\mathbf{W}} = \begin{pmatrix} \widehat{\mathbf{W}}_1 & 0 & \cdots & 0 \\ 0 & \widehat{\mathbf{W}}_2 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \widehat{\mathbf{W}}_{k(\lambda)} \end{pmatrix}, \quad \widehat{\boldsymbol{\Theta}} = \begin{pmatrix} \widehat{\boldsymbol{\Theta}}_1 & 0 & \cdots & 0 \\ 0 & \widehat{\boldsymbol{\Theta}}_2 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \widehat{\boldsymbol{\Theta}}_{k(\lambda)} \end{pmatrix} \tag{4.10}$$

Define the block diagonal matrices $\widehat{\mathbf{W}}_\ell$ or equivalently $\widehat{\boldsymbol{\Theta}}_\ell$ via the following sub-problems

$$\widehat{\boldsymbol{\Theta}}_\ell = \underset{\boldsymbol{\Theta}_\ell}{\arg\min} \quad \{-\log\det(\boldsymbol{\Theta}_\ell) + \operatorname{tr}(\mathrm{S}_\ell \boldsymbol{\Theta}_\ell) + \lambda \sum_{ij} |\, (\theta_\ell)_{ij} \,|\} \tag{4.11}$$

for $\ell = 1, \ldots, k(\lambda)$, where S_ℓ is a sub-block of S, with row/column indices from $\mathcal{V}_\ell^{(\lambda)} \times \mathcal{V}_\ell^{(\lambda)}$. The same notation is used for $\boldsymbol{\Theta}_\ell$. Denote the inverses of the block-precision matrices by $\{\widehat{\boldsymbol{\Theta}}_\ell\}^{-1} = \widehat{\mathbf{W}}_\ell$. We will show that the above $\widehat{\boldsymbol{\Theta}}$ satisfies the KKT conditions—(4.7) and

(4.8).

Note that by construction of the thresholded sample covariance graph,
if $i \in \mathcal{V}_\ell^{(\lambda)}$ and $j \in \mathcal{V}_{\ell'}^{(\lambda)}$ with $\ell \neq \ell'$, then $|s_{ij}| \leq \lambda$.

Hence, for $i \in \mathcal{V}_\ell^{(\lambda)}$ and $j \in \mathcal{V}_{\ell'}^{(\lambda)}$ with $\ell \neq \ell'$; the choice $\widehat{\theta}_{ij} = \widehat{w}_{ij} = 0$ satisfies the KKT conditions (4.7)

$$|s_{ij} - \widehat{w}_{ij}| \leq \lambda$$

for all the off-diagonal entries in the block-matrix (4.9).

By construction (4.11) it is easy to see that for every ℓ, the matrix $\widehat{\mathbf{\Theta}}_\ell$ satisfies the KKT conditions (4.7) and (4.8) corresponding to the ℓ^{th} block of the $p \times p$ dimensional problem. Hence $\widehat{\mathbf{\Theta}}$ solves problem (4.1).

The above argument shows that the connected components obtained from the estimated precision graph $\mathcal{G}^{(\lambda)}$ leads to a partition of the vertices $\{\widehat{\mathcal{V}}_\ell^{(\lambda)}\}_{1 \leq \ell \leq \kappa(\lambda)}$ such that for every $\ell \in \{1, \ldots, k(\lambda)\}$, there is a $\ell' \in \{1, \ldots, \kappa(\lambda)\}$ such that $\widehat{\mathcal{V}}_{\ell'}^{(\lambda)} \subset \mathcal{V}_\ell^{(\lambda)}$. In particular $k(\lambda) \leq \kappa(\lambda)$.

Conversely, if $\widehat{\mathbf{\Theta}}$ admits the decomposition as in the statement of the theorem, then it follows from (4.7) that:
for $i \in \widehat{\mathcal{V}}_\ell^{(\lambda)}$ and $j \in \widehat{\mathcal{V}}_{\ell'}^{(\lambda)}$ with $\ell \neq \ell'$; $|s_{ij} - \widehat{w}_{ij}| \leq \lambda$. Since $\widehat{w}_{ij} = 0$, we have $|s_{ij}| \leq \lambda$. This proves that the connected components of $\mathrm{G}^{(\lambda)}$ leads to a partition of the vertices, which is finer than the vertex-partition induced by the components of $\mathcal{G}^{(\lambda)}$. In particular this implies that $k(\lambda) \geq \kappa(\lambda)$.

Combining the above two we conclude $k(\lambda) = \kappa(\lambda)$ and also the equality (4.6). The permutation π in the theorem appears since the labeling of the connected components is not unique. $\square$

4.A.2 Proof of Theorem 10

Proof. This proof is a direct consequence of Theorem 9, which establishes that the vertex-partitions induced by the the connected components of the estimated precision graph and the thresholded sample covariance graph are equal.

Observe that, by construction, the connected components of the thresholded sample covariance graph i.e. $\mathrm{G}^{(\lambda)}$ are nested within the connected components of $\mathrm{G}^{(\lambda')}$. In particular, the vertex-partition induced by the components of the thresholded sample covariance graph at λ, is contained inside the vertex-partition induced by the components of the thresholded sample covariance graph at λ'. Now, using Theorem 9 we conclude that the vertex-partition induced by the components of the estimated precision graph at λ, given by $\{\widehat{\mathcal{V}}_\ell^{(\lambda)}\}_{1\leq\ell\leq\kappa(\lambda)}$ is contained inside the vertex-partition induced by the components of the estimated precision graph at λ', given by $\{\widehat{\mathcal{V}}_\ell^{(\lambda')}\}_{1\leq\ell\leq\kappa(\lambda')}$. The proof is thus complete. $\square$

Chapter 5

GLASSO New Insights and Alternatives

Abstract: The graphical lasso (Friedman, Hastie & Tibshirani 2007) is an algorithm for learning the structure in an undirected Gaussian graphical model, using ℓ_1 regularization to control the number of zeros in the precision matrix $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$ (Banerjee et al. 2008, Yuan & Lin 2007). The R package GLASSO (Friedman, Hastie & Tibshirani 2007) is popular, fast, and allows one to efficiently build a path of models for different values of the tuning parameter. Convergence of GLASSO can be tricky; the converged precision matrix might not be the inverse of the estimated covariance, and occasionally it fails to converge with warm starts. In this chapter we explain this behavior, and propose new algorithms that appear to outperform GLASSO.

By studying the "normal equations" we see that, GLASSO is solving the *dual* of the graphical lasso penalized likelihood, by block coordinate ascent; a result which can also be found in (Banerjee et al. 2008). In this dual, the target of estimation is $\mathbf{\Sigma}$, the covariance matrix, rather than the precision matrix $\mathbf{\Theta}$. We propose similar primal algorithms P-GLASSO and DP-GLASSO, that also operate by block-coordinate descent, where $\mathbf{\Theta}$ is the optimization target. We study all of these algorithms, and in particular different approaches to solving their coordinate sub-problems. We conclude that DP-GLASSO is superior from several points of view.

This chapter is based on joint work with Trevor Hastie and appears in our article Mazumder & Hastie (2011).

5.1 Introduction

We consider the same set-up as introduced in Chapter 4, where $\mathbf{X}_{n\times p}$, the data matrix denotes a a sample of n realizations from a p-dimensional Gaussian distribution with zero mean and positive definite covariance matrix $\boldsymbol{\Sigma}$. The focus of this chapter is to analyze the Graphical Lasso (Friedman, Hastie & Tibshirani 2007) algorithm and understand some of the apparent mysteries associated with it. Our algorithms focus either on the restricted version of $\boldsymbol{\Theta}$ or its inverse $\mathbf{W} = \boldsymbol{\Theta}^{-1}$. The graphical lasso problem minimizes a ℓ_1-regularized negative log-likelihood:

$$\underset{\boldsymbol{\Theta}\succ\mathbf{0}}{\text{minimize}}\, f(\boldsymbol{\Theta}) := -\log\det(\boldsymbol{\Theta}) + \operatorname{tr}(\mathrm{S}\boldsymbol{\Theta}) + \lambda\|\boldsymbol{\Theta}\|_1. \tag{5.1}$$

In this chapter we revisit the GLASSO algorithm proposed by Friedman, Hastie & Tibshirani (2007) for solving (5.1); we analyze its properties, expose problems and issues, and propose alternative algorithms more suitable for the task.

Some of the results and conclusions of this chapter can be found in (Banerjee et al. 2008), both explicitly and implicitly. We re-derive some of the results and derive new results, insights and algorithms, using a unified and more elementary framework.

Notation As in Chapter 4, we denote the entries of a matrix $\boldsymbol{A}_{n\times n}$ by a_{ij}. $\|\boldsymbol{A}\|_1$ denotes the sum of its absolute values, $\|\boldsymbol{A}\|_\infty$ the maximum absolute value of its entries, $\|\boldsymbol{A}\|_F$ is its Frobenius norm, and $\operatorname{abs}(\boldsymbol{A})$ is the matrix with elements $|a_{ij}|$. For a vector $\mathbf{u} \in \Re^q$, $\|\mathbf{u}\|_1$ denotes the ℓ_1 norm, and so on.

From now on, unless otherwise specified, we will assume that $\lambda > 0$.

5.2 Review of the GLASSO algorithm.

We use the frame-work of "normal equations" as in (Hastie et al. 2009, Friedman, Hastie & Tibshirani 2007). Using sub-gradient notation, we can write the optimality conditions (aka "normal equations") for a solution to (5.1) as

$$-\boldsymbol{\Theta}^{-1} + \mathbf{S} + \lambda\boldsymbol{\Gamma} = \mathbf{0}, \tag{5.2}$$

where $\boldsymbol{\Gamma}$ is a matrix of component-wise signs of $\boldsymbol{\Theta}$:

$$\begin{aligned} \gamma_{jk} &= \operatorname{sign}(\theta_{jk}) \text{ if } \theta_{jk} \neq 0 \\ \gamma_{jk} &\in [-1, 1] \text{ if } \theta_{jk} = 0 \end{aligned} \tag{5.3}$$

(we use the notation $\gamma_{jk} \in \operatorname{Sign}(\theta_{jk})$). Since the global stationary conditions of (5.2) require θ_{jj} to be positive, this implies that

$$w_{ii} = s_{ii} + \lambda,\ i = 1, \ldots, p, \tag{5.4}$$

where $\mathbf{W} = \boldsymbol{\Theta}^{-1}$.

GLASSO uses a block-coordinate method for solving (5.2). Consider a partitioning of $\boldsymbol{\Theta}$ and $\boldsymbol{\Gamma}$:

$$\boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{\Theta}_{11} & \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}_{21} & \theta_{22} \end{pmatrix}, \quad \boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{\Gamma}_{11} & \boldsymbol{\gamma}_{12} \\ \boldsymbol{\gamma}_{21} & \gamma_{22} \end{pmatrix} \tag{5.5}$$

where $\boldsymbol{\Theta}_{11}$ is $(p-1) \times (p-1)$, $\boldsymbol{\theta}_{12}$ is $(p-1) \times 1$ and θ_{22} is scalar. $\mathbf{W}$ and $\mathbf{S}$ are partitioned the same way. Using properties of inverses of block-partitioned matrices, observe that

$\mathbf{W} = \mathbf{\Theta}^{-1}$ can be written in two equivalent forms:

$$\begin{pmatrix} \mathbf{W}_{11} & \mathbf{w}_{12} \\ \mathbf{w}_{21} & w_{22} \end{pmatrix} = \begin{pmatrix} (\mathbf{\Theta}_{11} - \frac{\boldsymbol{\theta}_{12}\boldsymbol{\theta}_{21}}{\theta_{22}})^{-1} & -\mathbf{W}_{11}\frac{\boldsymbol{\theta}_{12}}{\theta_{22}} \\ \cdot & \frac{1}{\theta_{22}} - \frac{\boldsymbol{\theta}_{21}\mathbf{W}_{11}\boldsymbol{\theta}_{12}}{\theta_{22}^2} \end{pmatrix} \tag{5.6}$$

$$= \begin{pmatrix} \mathbf{\Theta}_{11}^{-1} + \frac{\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}\boldsymbol{\theta}_{21}\mathbf{\Theta}_{11}^{-1}}{(\theta_{22}-\boldsymbol{\theta}_{21}\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12})} & -\frac{\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}}{\theta_{22}-\boldsymbol{\theta}_{21}\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}} \\ \cdot & \frac{1}{(\theta_{22}-\boldsymbol{\theta}_{21}\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12})} \end{pmatrix}. \tag{5.7}$$

GLASSO solves for a row/column of (5.2) at a time, holding the rest fixed. Considering the pth column of (5.2), we get

$$-\mathbf{w}_{12} + \mathbf{s}_{12} + \lambda\boldsymbol{\gamma}_{12} = \mathbf{0}. \tag{5.8}$$

Reading off $\mathbf{w}_{12}$ from (5.6) we have

$$\mathbf{w}_{12} = -\mathbf{W}_{11}\boldsymbol{\theta}_{12}/\theta_{22} \tag{5.9}$$

and plugging into (5.8), we have:

$$\mathbf{W}_{11}\frac{\boldsymbol{\theta}_{12}}{\theta_{22}} + \mathbf{s}_{12} + \lambda\boldsymbol{\gamma}_{12} = \mathbf{0}. \tag{5.10}$$

GLASSO operates on the above gradient equation, as described below.

As a variation consider reading off $\mathbf{w}_{12}$ from (5.7):

$$\frac{\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}}{(\theta_{22} - \boldsymbol{\theta}_{21}\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12})} + \mathbf{s}_{12} + \lambda\boldsymbol{\gamma}_{12} = \mathbf{0}. \tag{5.11}$$

The above simplifies to

$$\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}w_{22} + \mathbf{s}_{12} + \lambda\boldsymbol{\gamma}_{12} = \mathbf{0}, \tag{5.12}$$

where $w_{22} = 1/(\theta_{22} - \boldsymbol{\theta}_{21}\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12})$ is fixed (by the global stationary conditions (5.4)). We will see that these two apparently similar estimating equations (5.10) and (5.12) lead to *very* different algorithms.

The GLASSO algorithm solves (5.10) for $\boldsymbol{\beta} = \boldsymbol{\theta}_{12}/\theta_{22}$, that is

$$\mathbf{W}_{11}\boldsymbol{\beta} + \mathbf{s}_{12} + \lambda\boldsymbol{\gamma}_{12} = \mathbf{0}, \tag{5.13}$$

where $\boldsymbol{\gamma}_{12} \in \text{Sign}(\boldsymbol{\beta})$, since $\theta_{22} > 0$. (5.13) is the stationarity equation for the following ℓ_1 regularized quadratic program:

$$\underset{\boldsymbol{\beta} \in \Re^{p-1}}{\text{minimize}} \quad \left\{ \tfrac{1}{2}\boldsymbol{\beta}'\mathbf{W}_{11}\boldsymbol{\beta} + \boldsymbol{\beta}'\mathbf{s}_{12} + \lambda\|\boldsymbol{\beta}\|_1 \right\}, \tag{5.14}$$

where $\mathbf{W}_{11} \succ 0$ is assumed to be fixed. This is analogous to a lasso regression problem of the last variable on the rest, except the cross-product matrix $\mathbf{S}_{11}$ is replaced by its current estimate $\mathbf{W}_{11}$. This problem itself can be solved efficiently using elementwise coordinate descent, exploiting the sparsity in $\boldsymbol{\beta}$. From $\hat{\boldsymbol{\beta}}$, it is easy to obtain $\hat{\mathbf{w}}_{12}$ from (5.9). Using the lower-right element of (5.6), $\hat{\theta}_{22}$ is obtained by

$$\frac{1}{\hat{\theta}_{22}} = w_{22} - \hat{\boldsymbol{\beta}}'\hat{\mathbf{w}}_{12}. \tag{5.15}$$

Finally, $\hat{\boldsymbol{\theta}}_{12}$ can now be recovered from $\hat{\boldsymbol{\beta}}$ and $\hat{\theta}_{22}$. Notice, however, that having solved for $\boldsymbol{\beta}$ and updated $\mathbf{w}_{12}$, GLASSO can move onto the next block; disentangling $\boldsymbol{\theta}_{12}$ and θ_{22} can be done at the end, when the algorithm over all blocks has converged. The GLASSO algorithm is outlined in Algorithm 4. We show in Lemma 14 in Section 5.8 that the successive updates in GLASSO keep $\mathbf{W}$ positive definite.

Figure 5.1 (left panel, black curve) plots the objective $f(\boldsymbol{\Theta}^{(k)})$ for the sequence of solutions produced by GLASSO on an example. Surprisingly, the curve is not monotone decreasing, as confirmed by the middle plot. If GLASSO were solving (5.1) by block coordinate-descent, we would not anticipate this behavior.

Algorithm 4 GLASSO algorithm

---

1. Initialize $\mathbf{W} = \mathbf{S} + \lambda\mathbf{I}$.
2. Cycle around the columns repeatedly, performing the following steps till convergence:
   (a) Rearrange the rows/columns so that the target column is last (implicitly).
   (b) Solve the lasso problem (5.14), using as warm starts the solution from the previous round for this column.
   (c) Update the row/column (off-diagonal) of the covariance using $\hat{\mathbf{w}}_{12}$ (5.9).
   (d) Save $\hat{\boldsymbol{\beta}}$ for this column in the matrix $\mathbf{B}$.
3. Finally, for every row/column, compute the diagonal entries $\hat{\theta}_{jj}$ using (5.15), and convert the $\mathbf{B}$ matrix to $\boldsymbol{\Theta}$.

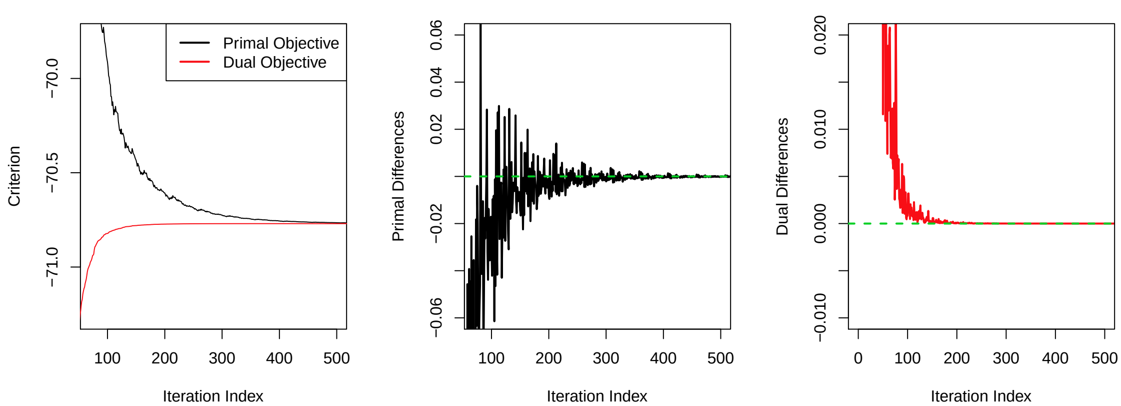

Figure 5.1: [Left panel] The objective values of the primal criterion (5.1) and the dual criterion (5.19) corresponding to the covariance matrix $\mathbf{W}$ produced by GLASSO algorithm as a function of the iteration index (each column/row update). [Middle Panel] The successive differences of the primal objective values — the zero crossings indicate non-monotonicity. [Right Panel] The successive differences in the dual objective values — there are no zero crossings, indicating that GLASSO produces a monotone sequence of dual objective values.

A closer look at steps (5.9) and (5.10) of the GLASSO algorithm leads to the following observations:

(a) We wish to solve (5.8) for $\boldsymbol{\theta}_{12}$. However $\boldsymbol{\theta}_{12}$ is entangled in $\mathbf{W}_{11}$, which is (incorrectly) treated as a constant.

(b) After updating $\boldsymbol{\theta}_{12}$, we see from (5.7) that the entire (working) covariance matrix $\mathbf{W}$ changes. GLASSO however updates only $\mathbf{w}_{12}$ and $\mathbf{w}_{21}$.

These two observations explain the non-monotone behavior of GLASSO in minimizing $f(\boldsymbol{\Theta})$. Section 5.3 shows a corrected block-coordinate descent algorithm for $\boldsymbol{\Theta}$, and Section 5.4 shows that the GLASSO algorithm is actually optimizing the dual of problem (5.1), with the optimization variable being $\mathbf{W}$.

5.3 A Corrected GLASSO block coordinate-descent algorithm

Recall that (5.12) is a variant of (5.10), where the dependence of the covariance sub-matrix $\mathbf{W}_{11}$ on $\boldsymbol{\theta}_{12}$ is explicit. With $\boldsymbol{\alpha} = \boldsymbol{\theta}_{12} w_{22}$ (with $w_{22} \geq 0$ fixed), $\boldsymbol{\Theta}_{11} \succ 0$, (5.12) is equivalent to the stationary condition for

$$\underset{\boldsymbol{\alpha} \in \Re^{p-1}}{\text{minimize}} \left\{ \tfrac{1}{2} \boldsymbol{\alpha}' \boldsymbol{\Theta}_{11}^{-1} \boldsymbol{\alpha} + \boldsymbol{\alpha}' \mathbf{s}_{12} + \lambda \|\boldsymbol{\alpha}\|_1 \right\}. \tag{5.16}$$

If $\hat{\boldsymbol{\alpha}}$ is the minimizer of (5.16), then $\hat{\boldsymbol{\theta}}_{12} = \hat{\boldsymbol{\alpha}}/w_{22}$. To complete the optimization for the entire row/column we need to update θ_{22}. This follows simply from (5.7)

$$\hat{\theta}_{22} = \frac{1}{w_{22}} + \hat{\boldsymbol{\theta}}_{21} \boldsymbol{\Theta}_{11}^{-1} \hat{\boldsymbol{\theta}}_{12}, \tag{5.17}$$

with $w_{22} = s_{22} + \lambda$.

To solve (5.16) we need $\boldsymbol{\Theta}_{11}^{-1}$ for each block update. We achieve this by maintaining $\mathbf{W} = \boldsymbol{\Theta}^{-1}$ as the iterations proceed. Then for each block

- we obtain $\mathbf{\Theta}_{11}^{-1}$ from

$$\mathbf{\Theta}_{11}^{-1} = \mathbf{W}_{11} - \mathbf{w}_{12}\mathbf{w}_{21}/w_{22}; \tag{5.18}$$

- once $\boldsymbol{\theta}_{12}$ is updated, the *entire* working covariance matrix $\mathbf{W}$ is updated (in particular the portions $\mathbf{W}_{11}$ and $\mathbf{w}_{12}$), via the identities in (5.7), using the known $\mathbf{\Theta}_{11}^{-1}$.

Both these steps are simple rank-one updates with a total cost of $O(p^2)$ operations.

We refer to this as the primal graphical lasso or P-GLASSO, which we present in Algorithm 5.

Algorithm 5 P-GLASSO Algorithm

1. Initialize $\mathbf{W} = \text{diag(S)} + \lambda\mathbf{I}$, and $\mathbf{\Theta} = \mathbf{W}^{-1}$.
2. Cycle around the columns repeatedly, performing the following steps till convergence:
   (a) Rearrange the rows/columns so that the target column is last (implicitly).
   (b) Compute $\mathbf{\Theta}_{11}^{-1}$ using (5.18).
   (c) Solve (5.16) for $\boldsymbol{\alpha}$, using as warm starts the solution from the previous round of row/column updates. Update $\hat{\boldsymbol{\theta}}_{12} = \hat{\boldsymbol{\alpha}}/w_{22}$, and $\hat{\theta}_{22}$ using (5.17).
   (d) Update $\mathbf{\Theta}$ and $\mathbf{W}$ using (5.7), ensuring that $\mathbf{\Theta}\mathbf{W} = \mathbf{I}_p$.
3. Output the solution $\mathbf{\Theta}$ (precision) and its exact inverse $\mathbf{W}$ (covariance).

The P-GLASSO algorithm requires slightly more work than GLASSO, since an additional $O(p^2)$ operations have to be performed before and after each block update. In return we have that after every row/column update, $\mathbf{\Theta}$ and $\mathbf{W}$ are positive definite (for $\lambda > 0$) and $\mathbf{\Theta}\mathbf{W} = \mathbf{I}_p$.

5.4 What is GLASSO actually solving?

Building upon the framework developed in Section 5.2, we now proceed to establish that GLASSO solves the convex dual of problem (5.1), by block coordinate ascent. We reach this conclusion via elementary arguments, closely aligned with the framework we develop in Section 5.2. The approach we present here is intended for an audience without much of a familiarity with convex duality theory (Boyd & Vandenberghe 2004).

Figure 5.1 illustrates that GLASSO is an ascent algorithm on the dual of the problem 5.1. The red curve in the left plot shows the dual objective rising monotonely, and the rightmost plot shows that the increments are indeed positive. There is an added twist though: in solving the block-coordinate update, GLASSO solves instead the dual of *that* subproblem.

5.4.1 Dual of the ℓ_1 regularized log-likelihood

We present below the following lemma, the conclusion of which also appears in (Banerjee et al. 2008), but we use the framework developed in Section 5.2.

Lemma 12. *Consider the primal problem (5.1) and its stationarity conditions (5.2). These are equivalent to the stationarity conditions for the box-constrained SDP*

$$\underset{\tilde{\boldsymbol{\Gamma}}:\, \|\tilde{\boldsymbol{\Gamma}}\|_\infty \leq \lambda}{\text{maximize}} \; g(\tilde{\boldsymbol{\Gamma}}) := \log\det(\mathrm{S} + \tilde{\boldsymbol{\Gamma}}) + p \tag{5.19}$$

under the transformation $\mathrm{S} + \tilde{\boldsymbol{\Gamma}} = \boldsymbol{\Theta}^{-1}$.

Proof. The (sub)gradient conditions (5.2) can be rewritten as:

$$-(\mathrm{S} + \lambda\boldsymbol{\Gamma})^{-1} + \boldsymbol{\Theta} = \mathbf{0} \tag{5.20}$$

where $\boldsymbol{\Gamma} = \mathrm{sgn}(\boldsymbol{\Theta})$. We write $\tilde{\boldsymbol{\Gamma}} = \lambda\boldsymbol{\Gamma}$ and observe that $\|\tilde{\boldsymbol{\Gamma}}\|_\infty \leq \lambda$. Denote by $\mathrm{abs}(\boldsymbol{\Theta})$ the matrix with element-wise absolute values.

Hence if $(\boldsymbol{\Theta}, \boldsymbol{\Gamma})$ satisfy (5.20), the substitutions

$$\tilde{\boldsymbol{\Gamma}} = \lambda\boldsymbol{\Gamma}; \quad \mathbf{P} = \mathrm{abs}(\boldsymbol{\Theta}) \tag{5.21}$$

satisfy the following set of equations:

$$\begin{aligned} -(\mathrm{S} + \tilde{\mathbf{\Gamma}})^{-1} + \mathbf{P} * \operatorname{sgn}(\tilde{\mathbf{\Gamma}}) &= \mathbf{0} \\ \mathbf{P} * (\operatorname{abs}(\tilde{\mathbf{\Gamma}}) - \lambda \mathbf{1}_p \mathbf{1}_p') &= \mathbf{0} \\ \|\tilde{\mathbf{\Gamma}}\|_\infty &\le \lambda. \end{aligned} \tag{5.22}$$

In the above, $\mathbf{P}$ is a symmetric $p \times p$ matrix with non-negative entries, $\mathbf{1}_p\mathbf{1}_p'$ denotes a $p \times p$ matrix of ones, and the operator '$*$' denotes element-wise product. We observe that (5.22) are the KKT optimality conditions for the box-constrained SDP (5.19). Similarly, the transformations $\mathbf{\Theta} = \mathbf{P} * \operatorname{sgn}(\tilde{\mathbf{\Gamma}})$ and $\mathbf{\Gamma} = \tilde{\mathbf{\Gamma}}/\lambda$ show that conditions (5.22) imply condition (5.20). Based on (5.20) the optimal solutions of the two problems (5.1) and (5.19) are related by $\mathrm{S} + \tilde{\mathbf{\Gamma}} = \mathbf{\Theta}^{-1}$. □

Notice that for the dual, the optimization variable is $\tilde{\mathbf{\Gamma}}$, with $\mathrm{S} + \tilde{\mathbf{\Gamma}} = \mathbf{\Theta}^{-1} = \mathbf{W}$. In other words, the dual problem solves for $\mathbf{W}$ rather than $\mathbf{\Theta}$, a fact that is suggested by the GLASSO algorithm.

Remark 2. *The equivalence of the solutions to problems (5.19) and (5.1) as described above can also be derived via convex duality theory (Boyd & Vandenberghe 2004), which shows that (5.19) is a dual function of the ℓ_1 regularized negative log-likelihood (5.1). Strong duality holds, hence the optimal solutions of the two problems coincide (Banerjee et al. 2008).*

We now consider solving (5.22) for the last block $\tilde{\boldsymbol{\gamma}}_{12}$ (excluding diagonal), holding the rest of $\tilde{\mathbf{\Gamma}}$ fixed. The corresponding equations are

$$\begin{aligned} -\boldsymbol{\theta}_{12} + \mathbf{p}_{12} * \operatorname{sgn}(\tilde{\boldsymbol{\gamma}}_{12}) &= \mathbf{0} \\ \mathbf{p}_{12} * (\operatorname{abs}(\tilde{\boldsymbol{\gamma}}_{12}) - \lambda \mathbf{1}_{p-1}) &= \mathbf{0} \\ \|\tilde{\boldsymbol{\gamma}}_{12}\|_\infty &\le \lambda. \end{aligned} \tag{5.23}$$

The only non-trivial translation is the $\boldsymbol{\theta}_{12}$ in the first equation. We must express this in terms of the optimization variable $\tilde{\boldsymbol{\gamma}}_{12}$. Since $\mathbf{s}_{12} + \tilde{\boldsymbol{\gamma}}_{12} = \mathbf{w}_{12}$, using the identities in (5.6), we have $\mathbf{W}_{11}^{-1}(\mathbf{s}_{12}+\tilde{\gamma}_{12}) = -\boldsymbol{\theta}_{12}/\theta_{22}$. Since $\theta_{22} > 0$, we can redefine $\tilde{\mathbf{p}}_{12} = \mathbf{p}_{12}/\theta_{22}$, to get

$$\begin{aligned}
\mathbf{W}_{11}^{-1}(\mathbf{s}_{12}+\tilde{\boldsymbol{\gamma}}_{12}) + \tilde{\mathbf{p}}_{12} * \mathrm{sgn}(\tilde{\boldsymbol{\gamma}}_{12}) &= \mathbf{0} \\
\tilde{\mathbf{p}}_{12} * (\mathrm{abs}(\tilde{\boldsymbol{\gamma}}_{12}) - \lambda \mathbf{1}_{p-1}) &= \mathbf{0} \\
\|\tilde{\boldsymbol{\gamma}}_{12}\|_\infty &\leq \lambda.
\end{aligned} \tag{5.24}$$

The following lemma shows that a block update of GLASSO solves (5.24) (and hence (5.23)), a block of stationary conditions for the dual of the graphical lasso problem. Curiously, GLASSO does this not directly, but by solving the dual of the QP corresponding to this block of equations.

Lemma 13. *Assume $\mathbf{W}_{11} \succ \mathbf{0}$. The stationarity equations*

$$\mathbf{W}_{11}\hat{\boldsymbol{\beta}} + \mathbf{s}_{12} + \lambda \hat{\boldsymbol{\gamma}}_{12} = 0, \tag{5.25}$$

where $\hat{\boldsymbol{\gamma}}_{12} \in Sign(\hat{\boldsymbol{\beta}})$, correspond to the solution of the ℓ_1-regularized QP:

$$\underset{\boldsymbol{\beta} \in \Re^{p-1}}{\text{minimize}} \ \tfrac{1}{2}\boldsymbol{\beta}'\mathbf{W}_{11}\boldsymbol{\beta} + \boldsymbol{\beta}'\mathbf{s}_{12} + \lambda\|\boldsymbol{\beta}\|_1. \tag{5.26}$$

Solving (5.26) is equivalent to solving the following box-constrained QP:

$$\underset{\boldsymbol{\gamma} \in \Re^{p-1}}{\text{minimize}} \ \tfrac{1}{2}(\mathbf{s}_{12}+\boldsymbol{\gamma})'\mathbf{W}_{11}^{-1}(\mathbf{s}_{12}+\boldsymbol{\gamma}) \ \textit{ subject to } \ \|\boldsymbol{\gamma}\|_\infty \leq \lambda, \tag{5.27}$$

with stationarity conditions given by (5.24), where the $\hat{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\gamma}}_{12}$ are related by

$$\hat{\boldsymbol{\beta}} = -\mathbf{W}_{11}^{-1}(\mathbf{s}_{12}+\tilde{\boldsymbol{\gamma}}_{12}). \tag{5.28}$$

Proof. (5.25) is the KKT optimality condition for the ℓ_1 regularized QP (5.26). We rewrite (5.25) as

$$\hat{\boldsymbol{\beta}} + \mathbf{W}_{11}^{-1}(\mathbf{s}_{12}+\lambda\hat{\boldsymbol{\gamma}}_{12}) = 0. \tag{5.29}$$

Observe that $\hat{\beta}_i = \text{sgn}(\hat{\beta}_i)|\beta_i| \ \forall i$ and $\|\hat{\boldsymbol{\gamma}}_{12}\|_\infty \leq 1$. Suppose $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\gamma}}_{12}$ satisfy (5.29), then the substitutions

$$\tilde{\boldsymbol{\gamma}}_{12} = \lambda\hat{\boldsymbol{\gamma}}_{12}, \quad \tilde{\mathbf{p}}_{12} = \text{abs}(\hat{\boldsymbol{\beta}}) \tag{5.30}$$

in (5.29) satisfy the stationarity conditions (5.24). It turns out that (5.24) is equivalent to the KKT optimality conditions of the box-constrained QP (5.27). Similarly, we note that if $\tilde{\boldsymbol{\gamma}}_{12}, \tilde{\mathbf{p}}_{12}$ satisfy (5.24), then the substitution

$$\hat{\boldsymbol{\gamma}}_{12} = \tilde{\boldsymbol{\gamma}}_{12}/\lambda; \quad \hat{\boldsymbol{\beta}} = \tilde{\mathbf{p}}_{12} * \text{sgn}(\tilde{\boldsymbol{\gamma}}_{12})$$

satisfies (5.29). Hence the $\hat{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\gamma}}_{12}$ are related by (5.28). □

Remark 3. *The above result can also be derived via convex duality theory(Boyd & Vandenberghe 2004), where (5.27) is actually the Lagrange dual of the ℓ_1 regularized QP (5.26), with (5.28) denoting the primal-dual relationship. (Banerjee et al. 2008, Section 3.3) interpret (5.27) as an ℓ_1 penalized regression problem (using convex duality theory) and explore connections with the set up of (Meinshausen & Bühlmann 2006).*

Note that the QP (5.27) is a (partial) optimization over the variable $\mathbf{w}_{12}$ only (since $\mathbf{s}_{12}$ is fixed); the sub-matrix $\mathbf{W}_{11}$ remains fixed in the QP. Exactly one row/column of $\mathbf{W}$ changes when the block-coordinate algorithm of GLASSO moves to a new row/column, unlike an explicit full matrix update in $\mathbf{W}_{11}$, which is required if $\boldsymbol{\theta}_{12}$ is updated. This again emphasizes that GLASSO is operating on the covariance matrix instead of $\boldsymbol{\Theta}$. We thus arrive at the following conclusion:

Theorem 11. GLASSO *performs block-coordinate ascent on the box-constrained SDP (5.19), the Lagrange dual of the primal problem (5.1). Each of the block steps are themselves box-constrained QPs, which* GLASSO *optimizes via their Lagrange duals.*

In our annotation perhaps GLASSO should be called DD-GLASSO, since it performs dual block updates for the dual of the graphical lasso problem. (Banerjee et al. 2008), the paper that inspired the original GLASSO article (Friedman, Hastie & Tibshirani 2007), also operates on the dual. They however solve the block-updates directly (which are box constrained QPs) using interior-point methods.

5.5 A New Algorithm — DP-GLASSO

In Section 5.3, we described P-GLASSO, a primal coordinate-descent method. For every row/column we need to solve a lasso problem (5.16), which operates on a quadratic form corresponding to the square matrix $\mathbf{\Theta}_{11}^{-1}$. There are two problems with this approach:

- the matrix $\mathbf{\Theta}_{11}^{-1}$ needs to be constructed at every row/column update with complexity $O(p^2)$;
- $\mathbf{\Theta}_{11}^{-1}$ is dense.

We now show how a simple modification of the ℓ_1-regularized QP leads to a box-constrained QP with attractive computational properties.

The KKT optimality conditions for (5.16), following (5.12), can be written as:

$$\mathbf{\Theta}_{11}^{-1}\boldsymbol{\alpha} + \mathbf{s}_{12} + \lambda\,\mathrm{sgn}(\boldsymbol{\alpha}) = 0. \tag{5.31}$$

Along the same lines of the derivations used in Lemma 13, the condition above is equivalent to

$$\begin{aligned} \tilde{\mathbf{q}}_{12} * \mathrm{sgn}(\tilde{\boldsymbol{\gamma}}) + \mathbf{\Theta}_{11}(\mathbf{s}_{12} + \tilde{\boldsymbol{\gamma}}) &= \mathbf{0} \\ \tilde{\mathbf{q}}_{12} * (\mathrm{abs}(\tilde{\boldsymbol{\gamma}}) - \lambda \mathbf{1}_{p-1}) &= 0 \\ \|\tilde{\boldsymbol{\gamma}}\|_\infty &\leq \lambda \end{aligned} \tag{5.32}$$

for some vector (with non-negative entries) $\tilde{\mathbf{q}}_{12}$. (5.32) are the KKT optimality conditions for the following box-constrained QP:

$$\underset{\boldsymbol{\gamma} \in \Re^{p-1}}{\text{minimize}} \; \tfrac{1}{2}(\mathbf{s}_{12} + \boldsymbol{\gamma})'\mathbf{\Theta}_{11}(\mathbf{s}_{12} + \boldsymbol{\gamma}); \;\; \text{subject to} \;\; \|\boldsymbol{\gamma}\|_\infty \leq \lambda. \tag{5.33}$$

The optimal solutions of (5.33) and (5.31) are related by

$$\hat{\boldsymbol{\alpha}} = -\boldsymbol{\Theta}_{11}(\mathbf{s}_{12} + \tilde{\boldsymbol{\gamma}}), \tag{5.34}$$

a consequence of (5.31), with $\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\theta}}_{12} \cdot w_{22}$ and $w_{22} = s_{22} + \lambda$. The diagonal θ_{22} of the precision matrix is updated via (5.7):

$$\hat{\theta}_{22} = \frac{1 - (\mathbf{s}_{12} + \tilde{\boldsymbol{\gamma}})' \hat{\boldsymbol{\theta}}_{12}}{w_{22}} \tag{5.35}$$

By strong duality, the box-constrained QP (5.33) with its optimality conditions (5.32) is equivalent to the lasso problem (5.16). Now both the problems listed at the beginning of the section are removed. The problem matrix $\boldsymbol{\Theta}_{11}$ is sparse, and no $O(p^2)$ updating is required after each block.

Algorithm 6 DP-GLASSO algorithm

1. Initialize $\boldsymbol{\Theta} = \text{diag}(\mathrm{S} + \lambda \mathbf{I})^{-1}$.
2. Cycle around the columns repeatedly, performing the following steps till convergence:
   (a) Rearrange the rows/columns so that the target column is last (implicitly).
   (b) Solve (5.33) for $\tilde{\boldsymbol{\gamma}}$ and update
   $$\hat{\boldsymbol{\theta}}_{12} = -\boldsymbol{\Theta}_{11}(\mathbf{s}_{12} + \tilde{\boldsymbol{\gamma}})/w_{22}$$
   (c) Solve for θ_{22} using (5.35).
   (d) Update the working covariance $\mathbf{w}_{12} = \mathbf{s}_{12} + \tilde{\boldsymbol{\gamma}}$.

The solutions returned at step 2(b) for $\hat{\boldsymbol{\theta}}_{12}$ need not be exactly sparse, even though it purports to produce the solution to the primal block problem (5.16), which is sparse. One needs to use a tight convergence criterion when solving (5.33). In addition, one can threshold those elements of $\hat{\boldsymbol{\theta}}_{12}$ for which $\tilde{\boldsymbol{\gamma}}$ is away from the box boundary, since those values are known to be zero.

Note that DP-GLASSO does to the primal formulation (5.1) what GLASSO does to the dual. DP-GLASSO operates on the precision matrix, whereas GLASSO operates on the covariance matrix.

5.6 Computational Costs in Solving the Block QPs

The ℓ_1 regularized QPs appearing in (5.14) and (5.16) are of the generic form

$$\underset{\mathbf{u}\in\Re^q}{\text{minimize}} \quad \tfrac{1}{2}\mathbf{u}'\mathbf{A}\mathbf{u} + \mathbf{a}'\mathbf{u} + \lambda\|\mathbf{u}\|_1, \tag{5.36}$$

for $\mathbf{A} \succ \mathbf{0}$. In this chapter, we choose to use cyclical coordinate descent for solving (5.36), as it is used in the GLASSO algorithm implementation of Friedman, Hastie & Tibshirani (2007). Moreover, cyclical coordinate descent methods perform well with good warm-starts. These are available for both (5.14) and (5.16), since they both maintain working copies of the precision matrix, updated after every row/column update. There are other efficient ways for solving (5.36), capable of scaling to large problems — for example first-order proximal methods (Beck & Teboulle 2009, Nesterov 2007), but we do not pursue them in this chapter.

The box-constrained QPs appearing in (5.27) and (5.33) are of the generic form:

$$\underset{\mathbf{v}\in\Re^q}{\text{minimize}} \quad \tfrac{1}{2}(\mathbf{v}+\mathbf{b})'\tilde{\mathbf{A}}(\mathbf{v}+\mathbf{b}) \;\; \text{subject to } \|\mathbf{v}\|_\infty \leq \lambda \tag{5.37}$$

for some $\tilde{\mathbf{A}} \succ \mathbf{0}$. As in the case above, we will use cyclical coordinate-descent for optimizing (5.37).

In general it is more efficient to solve (5.36) than (5.37) for larger values of λ. This is because a large value of λ in (5.36) results in sparse solutions $\hat{\mathbf{u}}$; the coordinate descent algorithm can easily detect when a zero stays zero, and no further work gets done for that coordinate on that pass. If the solution to (5.36) has κ non-zeros, then on average κ coordinates need to be updated. This leads to a cost of $O(q\kappa)$, for one full sweep across all the q coordinates.

On the other hand, a large λ for (5.37) corresponds to a weakly-regularized solution. Cyclical coordinate procedures for this task are not as effective. Every coordinate update of $\mathbf{v}$ results in updating the gradient, which requires adding a scalar multiple of a column of $\tilde{\mathbf{A}}$. If $\tilde{\mathbf{A}}$ is dense, this leads to a cost of $O(q)$, and for one full cycle across all the coordinates this costs $O(q^2)$, rather than the $O(q\kappa)$ for (5.36).

However, our experimental results show that DP-GLASSO is more efficient than GLASSO, so there are some other factors in play. When $\tilde{\mathbf{A}}$ is sparse, there are computational savings. If $\tilde{\mathbf{A}}$ has κq non-zeros, the cost per column reduces on average to $O(\kappa q)$ from $O(q^2)$. For the formulation (5.33) $\tilde{\mathbf{A}}$ is $\mathbf{\Theta}_{11}$, which is sparse for large λ. Hence for large λ, GLASSO and DP-GLASSO have similar costs.

For smaller values of λ, the box-constrained QP (5.37) is particularly attractive. Most of the coordinates in the optimal solution $\hat{\mathbf{v}}$ will pile up at the boundary points $\{-\lambda, \lambda\}$, which means that the coordinates need not be updated frequently. For problem (5.33) this number is also κ, the number of non-zero coefficients in the corresponding column of the precision matrix. If κ of the coordinates pile up at the boundary, then one full sweep of cyclical coordinate descent across all the coordinates will require updating gradients corresponding to the remaining $q - \kappa$ coordinates. Using similar calculations as before, this will cost $O(q(q-\kappa))$ operations per full cycle (since for small λ, $\tilde{\mathbf{A}}$ will be dense). For the ℓ_1 regularized problem (5.36), no such saving is achieved, and the cost is $O(q^2)$ per cycle.

Note that to solve problem (5.1), we need to solve a QP of a particular type (5.36) or (5.37) for a certain number of outer cycles (ie full sweeps across rows/columns). For every row/column update, the associated QP requires varying number of iterations to converge. It is hard to characterize all these factors and come up with precise estimates of convergence rates of the overall algorithm. However, we have observed that with warm-starts, on a relatively dense grid of λs, the complexities given above are pretty much accurate for DP-GLASSO (with warmstarts) specially when one is interested in solutions with small /

moderate accuracy. Our experimental results in Section 5.9.1 and Appendix Section 5.B support our observation.

We will now have a more critical look at the updates of the GLASSO algorithm and study their properties.

5.7 GLASSO: Positive definiteness, Sparsity and Exact Inversion

As noted earlier, GLASSO operates on $\mathbf{W}$ — it does *not* explicitly compute the inverse $\mathbf{W}^{-1}$. It does however keep track of the estimates for $\boldsymbol{\theta}_{12}$ after every row/column update. The copy of $\boldsymbol{\Theta}$ retained by GLASSO along the row/column updates is not the exact inverse of the optimization variable $\mathbf{W}$. Figure 5.2 illustrates this by plotting the squared-norm $\|(\boldsymbol{\Theta} - \mathbf{W}^{-1})\|_F^2$ as a function of the iteration index. Only upon (asymptotic) convergence, will $\boldsymbol{\Theta}$ be equal to $\mathbf{W}^{-1}$. This can have important consequences.

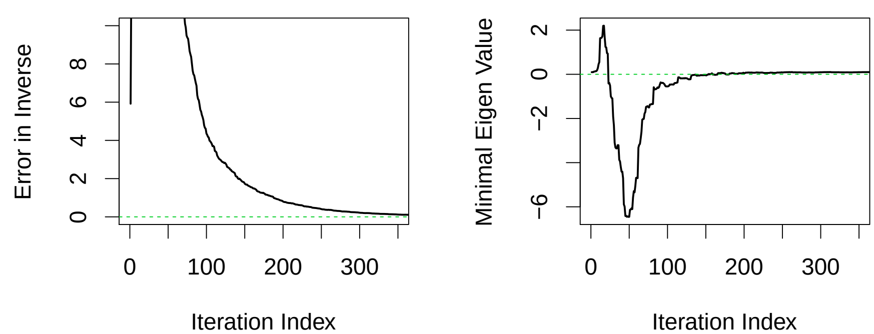

Figure 5.2: Figure illustrating some negative properties of GLASSO using a typical numerical example. [Left Panel] The precision matrix produced after every row/column update need not be the exact inverse of the working covariance matrix — the squared Frobenius norm of the error is being plotted across iterations. [Right Panel] The estimated precision matrix $\boldsymbol{\Theta}$ produced by GLASSO need not be positive definite along iterations; plot shows minimal eigen-value.

In many real-life problems one only needs an approximate solution to (5.1):

- for computational reasons it might be impractical to obtain a solution of high accuracy;
- from a statistical viewpoint it might be sufficient to obtain an approximate solution for $\mathbf{\Theta}$ that is *both* sparse and positive definite

It turns out that the GLASSO algorithm is not suited to this purpose.

Since the GLASSO is a block coordinate procedure on the covariance matrix, it maintains a positive definite covariance matrix at every row/column update. However, since the estimated precision matrix is not the exact inverse of $\mathbf{W}$, it need not be positive definite. Although it is relatively straightforward to maintain an exact inverse of $\mathbf{W}$ along the row/column updates (via simple rank-one updates as before), this inverse $\mathbf{W}^{-1}$ need *not* be sparse. Arbitrary thresholding rules may be used to set some of the entries to zero, but that might destroy the positive-definiteness of the matrix. Since a principal motivation of solving (5.1) is to obtain a sparse precision matrix (which is also positive definite), returning a dense $\mathbf{W}^{-1}$ to (5.1) is not desirable.

Figures 5.2 illustrates the above observations on a typical example.

The DP-GLASSO algorithm operates on the primal (5.1). Instead of optimizing the ℓ_1 regularized QP (5.16), which requires computing $\mathbf{\Theta}_{11}^{-1}$, DP-GLASSO optimizes (5.33). After every row/column update the precision matrix $\mathbf{\Theta}$ is positive definite. The working covariance matrix maintained by DP-GLASSO via $\mathbf{w}_{12} := \mathbf{s}_{12} + \hat{\gamma}$ need not be the exact inverse of $\mathbf{\Theta}$. Exact covariance matrix estimates, if required, can be obtained by tracking $\mathbf{\Theta}^{-1}$ via simple rank-one updates, as described earlier.

Unlike GLASSO, DP-GLASSO (and P-GLASSO) return a sparse and positive definite precision matrix even if the row/column iterations are terminated prematurely.

5.8 Warm Starts and Path-seeking Strategies

Since we seldom know in advance a good value of λ, we often compute a sequence of solutions to (5.1) for a (typically) decreasing sequence of values $\lambda_1 > \lambda_2 > \ldots > \lambda_K$. Warm-start or continuation methods use the solution at λ_i as an initial guess for the solution at λ_{i+1}, and often yield great efficiency. It turns out that for algorithms like GLASSO which operate on the dual problem, not all warm-starts necessarily lead to a convergent algorithm. We address this aspect in detail in this section.

The following lemma states the conditions under which the row/column updates of the GLASSO algorithm will maintain positive definiteness of the covariance matrix $\mathbf{W}$.

Lemma 14. *Suppose* $\mathbf{Z}$ *is used as a warm-start for the* GLASSO *algorithm. If* $\mathbf{Z} \succ \mathbf{0}$ *and* $\|\mathbf{Z} - \mathrm{S}\|_\infty \leq \lambda$, *then every row/column update of* GLASSO *maintains positive definiteness of the working covariance matrix* $\mathbf{W}$.

Proof. Recall that the GLASSO solves the dual (5.19). Assume $\mathbf{Z}$ is partitioned as in (5.5), and the pth row/column is being updated. Since $\mathbf{Z} \succ \mathbf{0}$, we have both

$$\mathbf{Z}_{11} \succ \mathbf{0} \text{ and } \left(z_{22} - \mathbf{z}_{21}(\mathbf{Z}_{11})^{-1}\mathbf{z}_{12}\right) > 0. \tag{5.38}$$

Since $\mathbf{Z}_{11}$ remains fixed, it suffices to show that after the row/column update, the expression $(\hat{w}_{22} - \hat{\mathbf{w}}_{21}(\mathbf{Z}_{11})^{-1}\hat{\mathbf{w}}_{12})$ remains positive. Recall that, via standard optimality conditions we have $\hat{w}_{22} = s_{22} + \lambda$, which makes $\hat{w}_{22} \geq z_{22}$ (since by assumption, $|z_{22} - s_{22}| \leq \lambda$ and $z_{22} > 0$). Furthermore, $\hat{\mathbf{w}}_{21} = \mathbf{s}_{21} + \hat{\boldsymbol{\gamma}}$, where $\hat{\boldsymbol{\gamma}}$ is the optimal solution to the corresponding box-QP (5.27). Since the starting solution $\mathbf{z}_{21}$ satisfies the box-constraint (5.27) i.e. $\|\mathbf{z}_{21} - \mathbf{s}_{21}\|_\infty \leq \lambda$, the optimal solution of the QP (5.27) improves the objective:

$$\hat{\mathbf{w}}_{21}(\mathbf{Z}_{11})^{-1}\hat{\mathbf{w}}_{12} \leq \mathbf{z}_{21}(\mathbf{Z}_{11})^{-1}\mathbf{z}_{12}$$

Combining the above along with the fact that $\hat{w}_{22} \geq z_{22}$ we see

$$\hat{w}_{22} - \hat{\mathbf{w}}_{21}(\mathbf{Z}_{11})^{-1}\hat{\mathbf{w}}_{12} > 0, \tag{5.39}$$

which implies that the new covariance estimate $\widehat{\mathbf{W}} \succ \mathbf{0}$. $\square$

Remark 4. *If the condition $\|\mathbf{Z} - \mathrm{S}\|_\infty \leq \lambda$ appearing in Lemma 14 is violated, then the row/column update of* GLASSO *need not maintain PD of the covariance matrix* $\mathbf{W}$.

We have encountered many counter-examples that show this to be true, see the discussion below.

The `R` package implementation of GLASSO allows the user to specify a warm-start as a tuple $(\mathbf{\Theta}_0, \mathbf{W}_0)$. This option is typically used in the construction of a path algorithm.

If $(\widehat{\mathbf{\Theta}}_\lambda, \widehat{\mathbf{W}}_\lambda)$ is provided as a warm-start for $\lambda' < \lambda$, then the GLASSO algorithm is not guaranteed to converge. It is easy to find numerical examples by choosing the gap $\lambda - \lambda'$ to be large enough. Among the various examples we encountered, we briefly describe one here. Details of the experiment/data and other examples can be found in the Appendix 5.A.1. We generated a data-matrix $\mathbf{X}_{n\times p}$, with $n = 2, p = 5$ with iid standard Gaussian entries. S is the sample covariance matrix. We solved problem (5.1) using GLASSO for $\lambda = 0.9 \times \max_{i\neq j} |s_{ij}|$. We took the estimated covariance and precision matrices: $\widehat{\mathbf{W}}_\lambda$ and $\widehat{\mathbf{\Theta}}_\lambda$ as a warm-start for the GLASSO algorithm with $\lambda' = \lambda \times 0.01$. The GLASSO algorithm failed to converge with this warm-start. We note that $\|\widehat{\mathbf{W}}_\lambda - \mathrm{S}\|_\infty = 0.0402 \nleq \lambda'$ (hence violating the sufficient condition in Lemma 15) and after updating the first row/column via the GLASSO algorithm we observed that "covariance matrix" $\mathbf{W}$ has negative eigen-values — leading to a non-convergent algorithm. The above phenomenon is not surprising and easy to explain and generalize. Since $\widehat{\mathbf{W}}_\lambda$ solves the dual (5.19), it is necessarily of the form $\widehat{\mathbf{W}}_\lambda = \mathrm{S} + \tilde{\mathbf{\Gamma}}$, for $\|\tilde{\mathbf{\Gamma}}\|_\infty \leq \lambda$. In the light of Lemma 14 and also Remark 4, the warm-start needs to be dual-feasible in order to guarantee that the iterates $\widehat{\mathbf{W}}$ remain PD and hence for the sub-problems to be well defined convex programs. Clearly $\widehat{\mathbf{W}}_\lambda$ does not satisfy the box-constraint $\|\widehat{\mathbf{W}}_\lambda - \mathrm{S}\|_\infty \leq \lambda'$, for $\lambda' < \lambda$. However, in practice the GLASSO algorithm is usually seen to converge (numerically) when λ' is quite *close* to λ.

The following lemma establishes that any PD matrix can be taken as a warm-start for

P-GLASSO or DP-GLASSOto ensure a convergent algorithm.

Lemma 15. *Suppose $\mathbf{\Phi} \succ \mathbf{0}$ is a used as a warm-start for the* P-GLASSO *(or* DP-GLASSO*) algorithm. Then every row/column update of* P-GLASSO *(or* DP-GLASSO*) maintains positive definiteness of the working precision matrix $\mathbf{\Theta}$.*

Proof. Consider updating the pth row/column of the precision matrix. The condition $\mathbf{\Phi} \succ \mathbf{0}$ is equivalent to both

$$\mathbf{\Phi}_{11} \succ \mathbf{0} \text{ and } \left(\phi_{22} - \mathbf{\Phi}_{21}(\mathbf{\Phi}_{11})^{-1}\mathbf{\Phi}_{12}\right) > 0.$$

Note that the block $\mathbf{\Phi}_{11}$ remains fixed; only the pth row/column of $\mathbf{\Theta}$ changes. $\boldsymbol{\phi}_{21}$ gets updated to $\hat{\boldsymbol{\theta}}_{21}$, as does $\hat{\boldsymbol{\theta}}_{12}$. From (5.7) the updated diagonal entry $\hat{\theta}_{22}$ satisfies:

$$\hat{\theta}_{22} - \hat{\boldsymbol{\theta}}_{21}(\mathbf{\Phi}_{11})^{-1}\hat{\boldsymbol{\theta}}_{12} = \frac{1}{(s_{22} + \lambda)} > 0.$$

Thus the updated matrix $\hat{\mathbf{\Theta}}$ remains PD. The result for the DP-GLASSO algorithm follows, since both the versions P-GLASSO and DP-GLASSO solve the same block coordinate problem. □

Remark 5. *A simple consequence of Lemmas 14 and 15 is that the QPs arising in the process, namely the ℓ_1 regularized QPs (5.14), (5.16) and the box-constrained QPs (5.27) and (5.33) are all valid convex programs, since all the respective matrices $\mathbf{W}_{11}$, $\mathbf{\Theta}_{11}^{-1}$ and $\mathbf{W}_{11}^{-1}$, $\mathbf{\Theta}_{11}$ appearing in the quadratic forms are PD.*

As exhibited in Lemma 15, both the algorithms DP-GLASSO and P-GLASSO are guaranteed to converge from any positive-definite warm start. This is due to the unconstrained formulation of the primal problem (5.1).

GLASSO really only requires an initialization for $\mathbf{W}$, since it constructs $\mathbf{\Theta}$ on the fly. Likewise DP-GLASSO only requires an initialization for $\mathbf{\Theta}$. Having the other half of the tuple assists in the block-updating algorithms. For example, GLASSO solves a series of lasso problems, where $\mathbf{\Theta}$ play the role as parameters. By supplying $\mathbf{\Theta}$ along with $\mathbf{W}$, the block-wise lasso problems can be given starting values close to the solutions. The same applies to DP-GLASSO. In neither case do the pairs have to be inverses of each other to serve this purpose.

If we wish to start with inverse pairs, and maintain such a relationship, we have described earlier how $O(p^2)$ updates after each block optimization can achieve this. One caveat for GLASSO is that starting with an inverse pair costs $O(p^3)$ operations, since we typically start with $\mathbf{W} = \mathrm{S} + \lambda\mathbf{I}$. For DP-GLASSO, we typically start with a diagonal matrix, which is trivial to invert.

5.9 Experimental Results & Timing Comparisons

We compared the performances of algorithms GLASSO and DP-GLASSO (both with and without warm-starts) on different examples with varying (n, p) values. While most of the results are presented in this section, some are relegated to the Appendix 5.B. Section 5.9.1 describes some synthetic examples and Section 5.9.2 presents comparisons on a real-life micro-array data-set.

5.9.1 Synthetic Experiments

In this section we present examples generated from two different covariance models — as characterized by the covariance matrix $\mathbf{\Sigma}$ or equivalently the precision matrix $\mathbf{\Theta}$. We create a data matrix $\mathbf{X}_{n\times p}$ by drawing n independent samples from a p dimensional normal distribution $\mathrm{MVN}(\mathbf{0}, \mathbf{\Sigma})$. The sample covariance matrix is taken as the input S to problem (5.1). The two covariance models are described below:

Type-1 The population concentration matrix $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$ has uniform sparsity with approximately 77 % of the entries zero.

We created the covariance matrix as follows. We generated a matrix $\mathbf{B}$ with iid standard Gaussian entries, symmetrized it via $\frac{1}{2}(\mathbf{B} + \mathbf{B}')$ and set approximately 77% of the entries of this matrix to zero, to obtain $\tilde{\mathbf{B}}$ (say). We added a scalar multiple of the p dimensional identity matrix to $\tilde{\mathbf{B}}$ to get the precision matrix $\mathbf{\Theta} = \tilde{\mathbf{B}} + \eta \mathbf{I}_{p\times p}$, with η chosen such that the minimum eigen value of $\mathbf{\Theta}$ is one.

Type-2 This example, taken from (Yuan & Lin 2007), is an auto-regressive process of order two — the precision matrix being tri-diagonal:

$$\theta_{ij} = \begin{cases} 0.5, & \text{if} \quad |j-i| = 1,\ i = 2, \ldots, (p-1); \\ 0.25, & \text{if} \quad |j-i| = 2,\ i = 3, \ldots, (p-2); \\ 1, & \text{if} \quad i = j,\ i = 1, \ldots, p; \text{and} \\ 0 & \text{otherwise.} \end{cases}$$

For each of the two set-ups Type-1 and Type-2 we consider twelve different combinations of (n, p):

(a) $p = 1000$, $n \in \{1500, 1000, 500\}$.

(b) $p = 800$, $n \in \{1000, 800, 500\}$.

(c) $p = 500$, $n \in \{800, 500, 200\}$.

(d) $p = 200$, $n \in \{500, 200, 50\}$.

For every (n, p) we solved (5.1) on a grid of twenty λ values linearly spaced in the log-scale, with $\lambda_i = 0.8^i \times \{0.9\lambda_{\max}\},\ i = 1, \ldots, 20$, where $\lambda_{\max} = \max_{i \neq j} |s_{ij}|$, is the off-diagonal entry of S with largest absolute value. $\lambda_{\max}$ is the smallest value of λ for which the solution to (5.1) is a diagonal matrix.

Since this article focuses on the GLASSO algorithm, its properties and alternatives that stem from the main idea of block-coordinate optimization, we present here the performances of the following algorithms:

Dual-Cold GLASSO with initialization $\mathbf{W} = \mathrm{S} + \lambda \mathbf{I}_{p\times p}$, as suggested in (Friedman, Hastie & Tibshirani 2007).

Dual-Warm The path-wise version of GLASSO with warm-starts, as suggested in (Friedman, Hastie & Tibshirani 2007). Although this path-wise version need not converge in general, this was not a problem in our experiments, probably due to the fine-grid of λ values.

Primal-Cold DP-GLASSO with diagonal initialization $\boldsymbol{\Theta} = (\mathrm{diag}(\mathrm{S}) + \lambda \mathbf{I})^{-1}$.

Primal-Warm The path-wise version of DP-GLASSO with warm-starts.

We did not include P-GLASSO in the comparisons above since P-GLASSO requires additional matrix rank-one updates after every row/column update, which makes it more expensive. None of the above listed algorithms require matrix inversions (via rank one updates). Furthermore, DP-GLASSO and P-GLASSO are quite similar as both are doing a block coordinate optimization on the dual. Hence we only included DP-GLASSO in our comparisons. We used our own implementation of the GLASSO and DP-GLASSO algorithm in R. The entire program is written in R, except the inner block-update solvers, which are the real work-horses:

- For GLASSO we used the lasso code crossProdLasso written in FORTRAN by (Friedman, Hastie & Tibshirani 2007);
- For DP-GLASSO we wrote our own FORTRAN code to solve the box QP.

An R package implementing DP-GLASSO will be made available in CRAN.

In the figure and tables that follow below, for every algorithm, at a fixed λ we report the *total time* taken by *all* the QPs — the ℓ_1 regularized QP for GLASSO and the box constrained QP for DP-GLASSO till convergence All computations were done on a Linux machine with model specs: Intel(R) Xeon(R) CPU 5160 @ 3.00GHz.

Convergence Criterion: Since DP-GLASSO operates on the the primal formulation and GLASSO operates on the dual — to make the convergence criteria comparable across examples we based it on the relative change in the primal objective values i.e. $f(\mathbf{\Theta})$ (5.1) across two successive iterations:

$$\frac{f(\mathbf{\Theta}_k) - f(\mathbf{\Theta}_{k-1})}{|f(\mathbf{\Theta}_{k-1})|} \leq \text{TOL}, \tag{5.40}$$

where one iteration refers to a full sweep across p rows/columns of the precision matrix (for DP-GLASSO) and covariance matrix (for GLASSO); and TOL denotes the tolerance level or level of accuracy of the solution. To compute the primal objective value for the GLASSO algorithm, the precision matrix is computed from $\widehat{\mathbf{W}}$ via direct inversion (the time taken for inversion and objective value computation is not included in the timing comparisons).

Computing the objective function is quite expensive relative to the computational cost of the iterations. In our experience convergence criteria based on a relative change in the precision matrix for DP-GLASSO and the covariance matrix for GLASSO seemed to be a practical choice for the examples we considered. However, for reasons we described above, we used criterion 5.40 in the experiments.

Observations: Figure 5.4 presents the times taken by the algorithms to converge to an accuracy of $\text{TOL} = 10^{-4}$ on a grid of λ values.

The figure shows eight different scenarios with $p > n$, corresponding to the two different covariance models Type-1 (left panel) and Type-2 (right panel). It is quite evident that DP-GLASSO with warm-starts (Primal-Warm) outperforms all the other algorithms across all the different examples. All the algorithms converge quickly for large values of λ (typically high sparsity) and become slower with decreasing λ. For large p and small λ, convergence is slow; however for $p > n$, the non-sparse end of the regularization path is really not that interesting from a statistical viewpoint. Warm-starts apparently do *not* always help in speeding up the convergence of GLASSO ; for example see Figure 5.4 with $(n, p) = (500, 1000)$ (Type 1) and $(n, p) = (500, 800)$ (Type 2). This probably further validates the fact that warm-starts in the case of GLASSO need to be carefully designed, in order for them to *speed-up* convergence. Note however, that GLASSO with the warm-starts prescribed is not even guaranteed to converge — we however did not come across any such instance among the experiments presented in this section.

Based on the suggestion of a referee we annotated the plots in Figure 5.4 with locations in the regularization path that are of interest. For each plot, two vertical dotted lines are drawn which correspond to the λs at which the distance of the estimated precision matrix $\widehat{\mathbf{\Theta}}_\lambda$ from the population precision matrix is minimized wrt to the $\|\cdot\|_1$ norm (green) and $\|\cdot\|_F$ norm (blue). The optimal λ corresponding to the $\|\cdot\|_1$ metric chooses sparser models than those chosen by $\|\cdot\|_F$; the performance gains achieved by DP-GLASSO seem to be more prominent for the latter λ.

Table 5.1 presents the timings for all the four algorithmic variants on the twelve different (n, p) combinations listed above for Type 1. For every example, we report the total time till convergence on a grid of twenty λ values for two different tolerance levels: TOL $\in \{10^{-4}, 10^{-5}\}$. Note that the DP-GLASSO returns positive definite and sparse precision matrices even if the algorithm is terminated at a relatively small/moderate accuracy level — this is not the case in GLASSO . The rightmost column presents the proportion of non-zeros averaged across the entire path of solutions $\widehat{\mathbf{\Theta}}_\lambda$, where $\widehat{\mathbf{\Theta}}_\lambda$ is obtained by solving (5.1) to a high precision i.e. 10^{-6}, by algorithms GLASSO and DP-GLASSO and averaging the results.

Again we see that in all the examples DP-GLASSO with warm-starts is the clear winner among its competitors. For a fixed p, the total time to trace out the path generally decreases with increasing n. There is no clear winner between GLASSO with warm-starts and GLASSO without warm-starts. It is often seen that DP-GLASSO without warm-starts converges faster than both the variants of GLASSO (with and without warm-starts).

Table 5.2 reports the timing comparisons for Type 2. Once again we see that in all the examples Primal-Warm turns out to be the clear winner.

For $n \leq p = 1000$, we observe that Primal-Warm is generally faster for Type-2 than Type-1. This however, is reversed for smaller values of $p \in \{800, 500\}$. Primal-Cold is has a smaller overall computation time for Type-1 over Type-2. In some cases (for example $n \leq p = 1000$), we see that Primal-Warm in Type-2 converges much faster than its competitors on a relative scale than in Type-1 — this difference is due to the variations in the structure of the covariance matrix.

p / n	relative error (TOL)	Total time (secs) to compute a path of solutions				Average % Zeros in path
		Dual-Cold	Dual-Warm	Primal-Cold	Primal-Warm	
1000 / 500	10^{-4}	3550.71	6592.63	2558.83	**2005.25**	80.2
	10^{-5}	4706.22	8835.59	3234.97	**2832.15**	
1000 / 1000	10^{-4}	2788.30	3158.71	2206.95	**1347.05**	83.0
	10^{-5}	3597.21	4232.92	2710.34	**1865.57**	
1000 / 1500	10^{-4}	2447.19	4505.02	1813.61	**932.34**	85.6
	10^{-5}	2764.23	6426.49	2199.53	**1382.64**	
800 / 500	10^{-4}	1216.30	2284.56	928.37	**541.66**	78.8
	10^{-5}	1776.72	3010.15	1173.76	**798.93**	
800 / 800	10^{-4}	1135.73	1049.16	788.12	**438.46**	80.0
	10^{-5}	1481.36	1397.25	986.19	**614.98**	
800 / 1000	10^{-4}	1129.01	1146.63	786.02	**453.06**	80.2
	10^{-5}	1430.77	1618.41	992.13	**642.90**	
500 / 200	10^{-4}	605.45	559.14	395.11	**191.88**	75.9
	10^{-5}	811.58	795.43	520.98	**282.65**	
500 / 500	10^{-4}	427.85	241.90	252.83	**123.35**	75.2
	10^{-5}	551.11	315.86	319.89	**182.81**	
500 / 800	10^{-4}	359.78	279.67	207.28	**111.92**	80.9
	10^{-5}	416.87	402.61	257.06	**157.13**	
200 / 50	10^{-4}	65.87	50.99	37.40	**23.32**	75.6
	10^{-5}	92.04	75.06	45.88	**35.81**	
200 / 200	10^{-4}	35.29	25.70	17.32	**11.72**	66.8
	10^{-5}	45.90	33.23	22.41	**17.16**	
200 / 300	10^{-4}	32.29	23.60	16.30	**10.77**	66.0
	10^{-5}	38.37	33.95	20.12	**15.12**	

Table 5.1: Table showing the performances of the four algorithms GLASSO (Dual-Warm/Cold) and DP-GLASSO (Primal-Warm/Cold) for the covariance model Type-1. We present the times (in seconds) required to compute a path of solutions to (5.1) (on a grid of twenty λ values) for different (n, p) combinations and relative errors (as in (5.40)). The rightmost column gives the averaged sparsity level across the grid of λ values. DP-GLASSO with warm-starts is consistently the winner across all the examples.

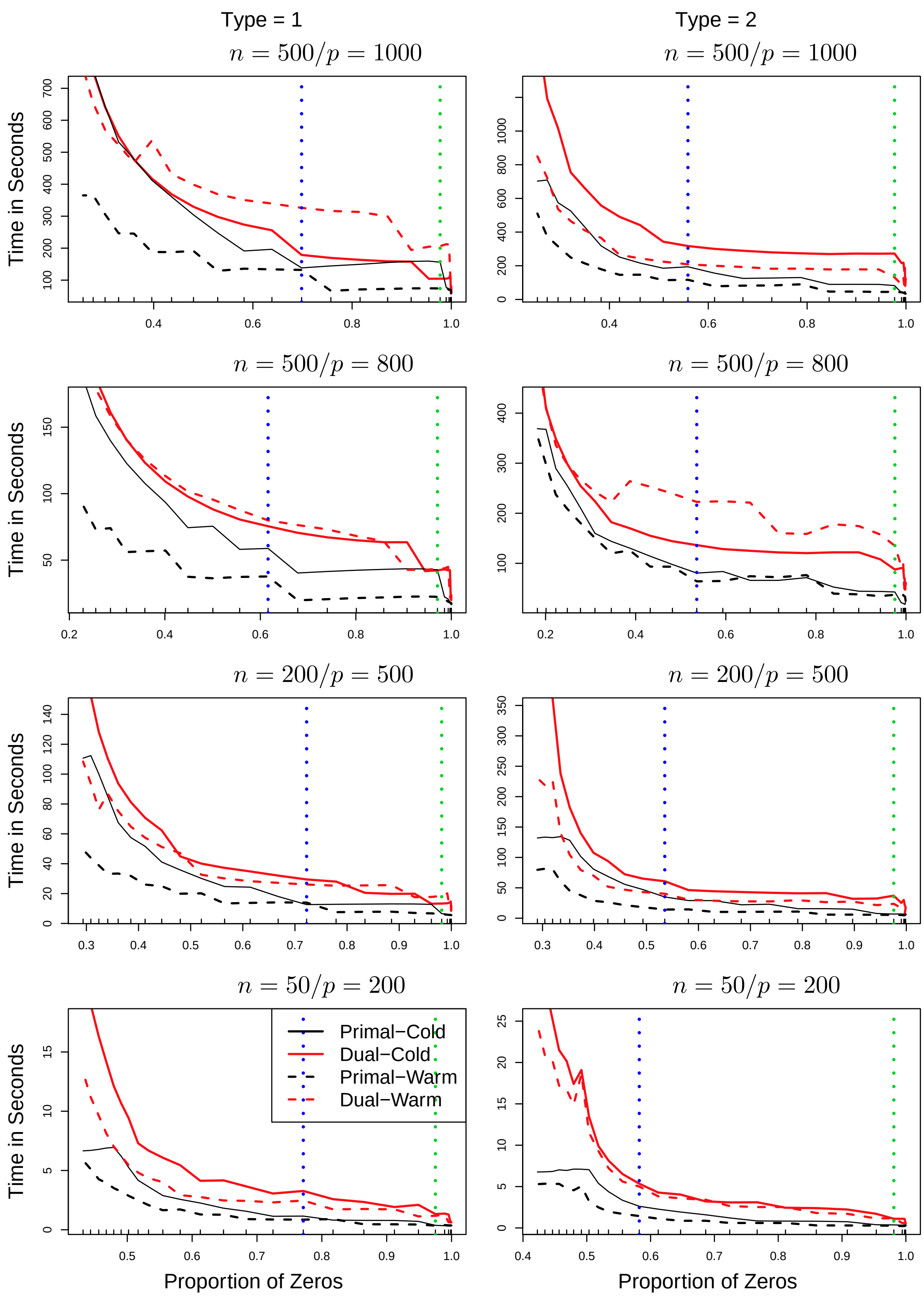

Figure 5.3: The timings in seconds for the four different algorithmic versions: GLASSO (with and without warm-starts) and DP-GLASSO (with and without warm-starts) for a grid of λ values on the log-scale. [Left Panel] Covariance model for Type-1, [Right Panel] Covariance model for Type-2. The horizontal axis is indexed by the proportion of zeros in the solution. The vertical dashed lines correspond to the optimal λ values for which the estimated errors $\|\widehat{\mathbf{\Theta}}_\lambda - \mathbf{\Theta}\|_1$ (green) and $\|\widehat{\mathbf{\Theta}}_\lambda - \mathbf{\Theta}\|_F$ (blue) are minimum.

p / n	relative error (TOL)	Total time (secs) to compute a path of solutions Dual-Cold	Dual-Warm	Primal-Cold	Primal-Warm	Average % Zeros in path
1000 / 500	10^{-4}	6093.11	5483.03	3495.67	**1661.93**	75.6
	10^{-5}	7707.24	7923.80	4401.28	**2358.08**	
1000 / 1000	10^{-4}	4773.98	3582.28	2697.38	**1015.84**	76.70
	10^{-5}	6054.21	4714.80	3444.79	**1593.54**	
1000 / 1500	10^{-4}	4786.28	5175.16	2693.39	**1062.06**	78.5
	10^{-5}	6171.01	6958.29	3432.33	**1679.16**	
800 / 500	10^{-4}	2914.63	3466.49	1685.41	**1293.18**	74.3
	10^{-5}	3674.73	4572.97	2083.20	**1893.22**	
800 / 800	10^{-4}	2021.55	1995.90	1131.35	**618.06**	74.4
	10^{-5}	2521.06	2639.62	1415.95	**922.93**	
800 / 1000	10^{-4}	3674.36	2551.06	1834.86	**885.79**	75.9
	10^{-5}	4599.59	3353.78	2260.58	**1353.28**	
500 / 200	10^{-4}	1200.24	885.76	718.75	**291.61**	70.5
	10^{-5}	1574.62	1219.12	876.45	**408.41**	
500 / 500	10^{-4}	575.53	386.20	323.30	**130.59**	72.2
	10^{-5}	730.54	535.58	421.91	**193.08**	
500 / 800	10^{-4}	666.75	474.12	373.60	**115.75**	73.7
	10^{-5}	852.54	659.58	485.47	**185.60**	
200 / 50	10^{-4}	110.18	98.23	48.98	**26.97**	73.0
	10^{-5}	142.77	133.67	55.27	**33.95**	
200 / 200	10^{-4}	50.63	40.68	23.94	**9.97**	63.7
	10^{-5}	66.63	56.71	31.57	**14.70**	
200 / 300	10^{-4}	47.63	36.18	21.24	**8.19**	65.0
	10^{-5}	60.98	50.52	27.41	**12.22**	

Table 5.2: Table showing comparative timings of the four algorithmic variants of GLASSO and DP-GLASSO for the covariance model in Type-2. This table is similar to Table 5.1, displaying results for Type-1. DP-GLASSO with warm-starts consistently outperforms all its competitors.

5.9.2 Micro-array Example

We consider the data-set introduced in (Alon et al. 1999) and further studied in Rothman et al. (2008) and Mazumder & Hastie (2012). In this experiment, tissue samples were analyzed using an Affymetrix Oligonucleotide array. The data was processed, filtered and reduced to a subset of 2000 gene expression values. The number of Colon Adenocarcinoma tissue samples is $n = 62$. For the purpose of the experiments presented in this section, we pre-screened the genes to a size of $p = 725$. We obtained this subset of genes using the idea of *exact covariance thresholding* introduced in our paper (Mazumder & Hastie 2012). We thresholded the sample correlation matrix obtained from the 62×2000 microarray data-matrix into connected components with a threshold of 0.00364[1] — the genes belonging to the largest connected component formed our pre-screened gene pool of size $p = 725$. This (subset) data-matrix of size $(n, p) = (62, 725)$ is used for our experiments.

The results presented below in Table 5.3 show timing comparisons of the four different algorithms: Primal-Warm/Cold and Dual-Warm/Cold on a grid of fifteen λ values in the log-scale. Once again we see that the Primal-Warm outperforms the others in terms of speed and accuracy. Dual-Warm performs quite well in this example.

relative	Total time (secs) to compute a path of solutions			
error (TOL)	Dual-Cold	Dual-Warm	Primal-Cold	Primal-Warm
10^{-3}	515.15	406.57	462.58	**334.56**
10^{-4}	976.16	677.76	709.83	**521.44**

Table 5.3: Comparisons among algorithms for a microarray dataset with $n = 62$ and $p = 725$, for different tolerance levels (TOL). We took a grid of fifteen λ values, the average % of zeros along the whole path is 90.8.

[1]this is the largest value of the threshold for which the size of the largest connected component is smaller than 800

5.10 Conclusions

This chapter explores some of the apparent mysteries in the behavior of the GLASSO algorithm introduced in (Friedman, Hastie & Tibshirani 2007). These have been explained by leveraging the fact that the GLASSO algorithm is solving the dual of the graphical lasso problem (5.1), by block coordinate ascent. Each block update, itself the solution to a convex program, is solved via its own dual, which is equivalent to a lasso problem. The optimization variable is $\mathbf{W}$, the covariance matrix, rather than the target precision matrix $\mathbf{\Theta}$. During the course of the iterations, a working version of $\mathbf{\Theta}$ is maintained, but it may not be positive definite, and its inverse is not $\mathbf{W}$. Tight convergence is therefore essential, for the solution $\hat{\mathbf{\Theta}}$ to be a proper inverse covariance. There are issues using warm starts with GLASSO, when computing a path of solutions. Unless the sequence of λs are sufficiently close, since the "warm start"s are not dual feasible, the algorithm can get into trouble.

We have also developed two primal algorithms P-GLASSO and DP-GLASSO. The former is more expensive, since it maintains the relationship $\mathbf{W} = \mathbf{\Theta}^{-1}$ at every step, an $O(p^3)$ operation per sweep across all row/columns. DP-GLASSO is similar in flavor to GLASSO except its optimization variable is $\mathbf{\Theta}$. It also solves the dual problem when computing its block update, in this case a box-QP. This box-QP has attractive sparsity properties at *both* ends of the regularization path, as evidenced in some of our experiments. It maintains a positive definite $\mathbf{\Theta}$ throughout its iterations, and can be started at any positive definite matrix. Our experiments show in addition that DP-GLASSO is faster than GLASSO.

An R package implementing DP-GLASSO will be made available in CRAN.

5.A Appendix to Chapter 5

This section complements the examples provided in the chapter with further experiments and illustrations.

5.A.1 Examples: Non-Convergence of GLASSO with warm-starts

This section illustrates with examples that warm-starts for the GLASSO need not converge. This is a continuation of examples presented in Section 5.8.

Example 1:

We took $(n, p) = (2, 5)$ and setting the seed of the random number generator in R as set.seed(2008) we generated a data-matrix $\mathbf{X}_{n \times p}$ with iid standard Gaussian entries. The sample covariance matrix S is given below:

$$\begin{pmatrix} 0.03597652 & 0.03792221 & 0.1058585 & -0.08360659 & 0.1366725 \\ 0.03597652 & 0.03792221 & 0.1058585 & -0.08360659 & 0.1366725 \\ 0.10585853 & 0.11158361 & 0.3114818 & -0.24600689 & 0.4021497 \\ -0.08360659 & -0.08812823 & -0.2460069 & 0.19429514 & -0.3176160 \\ 0.13667246 & 0.14406402 & 0.4021497 & -0.31761603 & 0.5192098 \end{pmatrix}$$

With q denoting the maximum off-diagonal entry of S (in absolute value), we solved (5.1) using GLASSO at $\lambda = 0.9 \times q$. The covariance matrix for this λ was taken as a warm-start for the GLASSO algorithm with $\lambda' = \lambda \times 0.01$. The smallest eigen-value of the working covariance matrix $\mathbf{W}$ produced by the GLASSO algorithm, upon updating the first row/column was: -0.002896128, which is clearly undesirable for the convergence of the algorithm GLASSO . This is why the algorithm GLASSO breaks down.

Example 2:

The example is similar to above, with $(n, p) = (10, 50)$, the seed of random number generator in R being set to set.seed(2008) and $\mathbf{X}_{n \times p}$ is the data-matrix with iid Gaussian entries. If the covariance matrix $\widehat{\mathbf{W}_\lambda}$ which solves problem (5.1) with $\lambda = 0.9 \times \max_{i \neq j} |s_{ij}|$ is taken as a warm-start to the GLASSO algorithm with $\lambda' = \lambda \times 0.1$ — the algorithm fails to converge. Like the previous example, after the first row/column update, the working covariance matrix has negative eigen-values.

5.B Further Experiments and Numerical Studies

This section is a continuation to Section 5.9, in that it provides further examples comparing the performance of algorithms GLASSO and DP-GLASSO . The experimental data is generated as follows. For a fixed value of p, we generate a matrix $\mathbf{A}_{p\times p}$ with random Gaussian entries. The matrix is symmetrized by $\mathbf{A} \leftarrow (\mathbf{A} + \mathbf{A}')/2$. Approximately half of the off-diagonal entries of the matrix are set to zero, uniformly at random. All the eigen-values of the matrix $\mathbf{A}$ are lifted so that the smallest eigen-value is zero. The noiseless version of the precision matrix is given by $\mathbf{\Theta} = \mathbf{A} + \tau \mathbf{I}_{p\times p}$. We generated the sample covariance matrix S by adding symmetric positive semi-definite random noise $\mathbf{N}$ to $\mathbf{\Theta}^{-1}$; i.e. $\mathrm{S} = \mathbf{\Theta}^{-1} + \mathbf{N}$, where this noise is generated in the same manner as $\mathbf{A}$. We considered four different values of $p \in \{300, 500, 800, 1000\}$ and two different values of $\tau \in \{1, 4\}$.

For every p, τ combination we considered a path of twenty λ values on the geometric scale. For every such case four experiments were performed: Primal-Cold, Primal-Warm, Dual-Cold and Dual-Warm (as described in Section 5.9). Each combination was run 5 times, and the results averaged, to avoid dependencies on machine loads. Figure 5.4 shows the results. Overall, DP-GLASSO with warm starts performs the best, especially at the extremes of the path. We gave some explanation for this in Section 5.6. For the largest problems ($p = 1000$) their performances are comparable in the central part of the path (though DP-GLASSO dominates), but at the extremes DP-GLASSO dominates by a large margin.

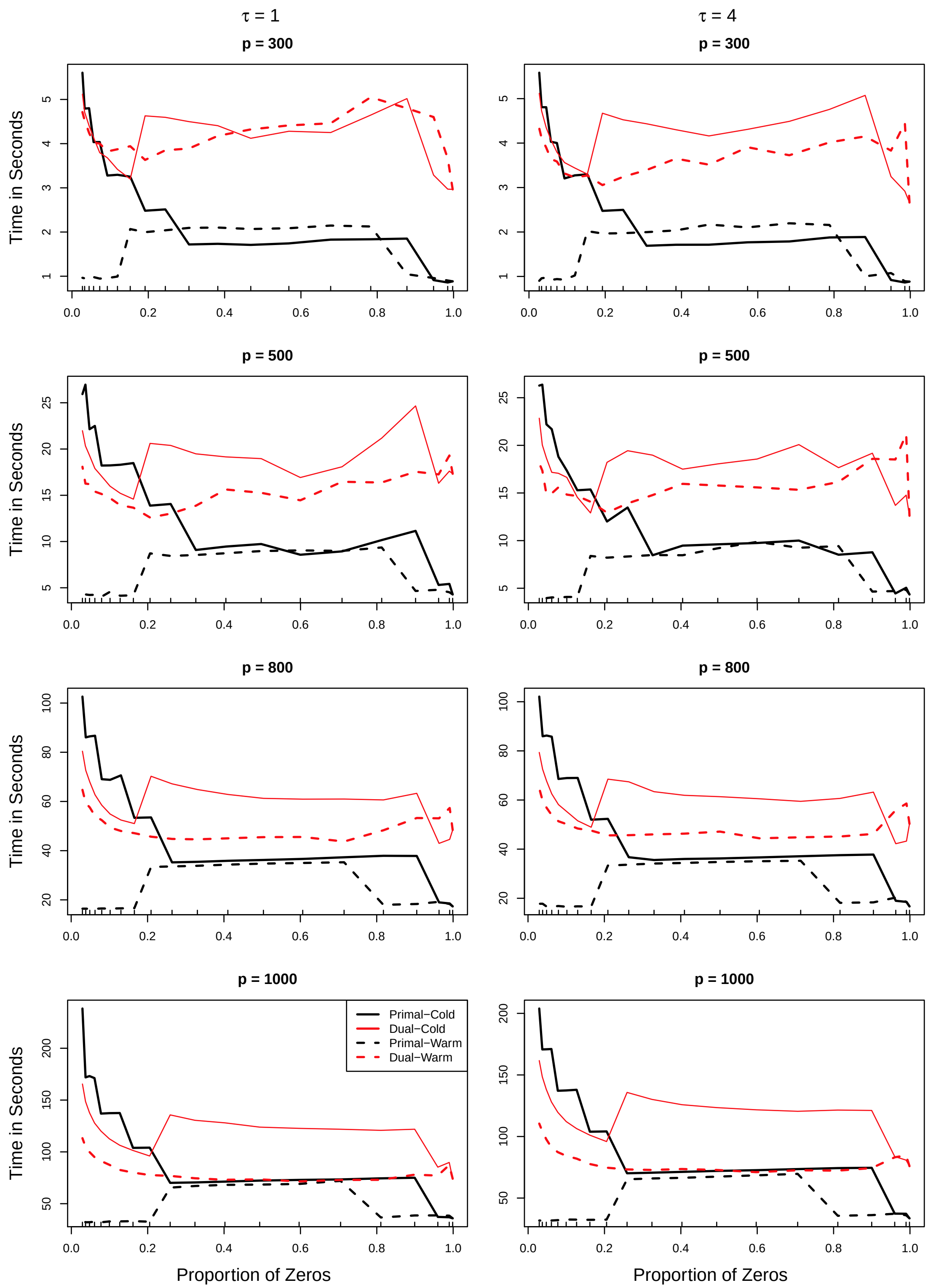

Figure 5.4: The timings in seconds for the four different algorithmic versions GLASSO (with and without warm-starts) and DP-GLASSO (with and without warm-starts) for a grid of twenty λ values on the log-scale. The horizontal axis is indexed by the proportion of zeros in the solution.

Bibliography

Abernethy, J., Bach, F., Evgeniou, T. & Vert, J.-P. (2009), 'A new approach to collaborative filtering: operator estimation with spectral regularization', *Journal of Machine Learning Research* **10**, 803–826.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D. & Levine, A. J. (1999), 'Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays', *Proceedings of the National Academy of Sciences of the United States of America* **96**(12), 6745–6750.
URL: *http://dx.doi.org/10.1073/pnas.96.12.6745*

Argyriou, A., Evgeniou, T. & Pontil, M. (2007), Multi-task feature learning, *in* 'Advances in Neural Information Processing Systems 19', MIT Press.

Argyriou, A., Evgeniou, T. & Pontil, M. (2008), 'Convex multi-task feature learning', *Machine Learning* **73(3)**, 243–272.

Bach, F. (2008), 'Consistency of trace norm minimization', *Journal of Machine Learning Research* **9**, 1019–1048.

Banerjee, O., Ghaoui, L. E. & d'Aspremont, A. (2008), 'Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data', *Journal of Machine Learning Research* **9**, 485–516.

Beck, A. & Teboulle, M. (2009), 'A fast iterative shrinkage-thresholding algorithm for linear inverse problems', *SIAM J. Imaging Sciences* **2**(1), 183–202.

Bell, R. M. & Koren., Y. (2007), Lessons from the Netflix prize challenge, Technical report, AT&T Bell Laboratories.

Bollobas, B. (1998), *Modern graph theory*, Springer, New York.

Borwein, J. & Lewis, A. (2006), *Convex Analysis and Nonlinear Optimization*, Springer.

Boyd, S. & Vandenberghe, L. (2004), *Convex Optimization*, Cambridge University Press.

Breiman, L. (1996), 'Heuristics of instability and stabilization in model selection', *Annals of Statistics* **24**, 2350–2383.

Burer, S. & Monteiro, R. D. (2005), 'Local minima and convergence in low-rank semidefinite programming', *Mathematical Programming* **103(3)**, 427–631.

Cai, J.-F., Candès, E. J. & Shen, Z. (2010), 'A singular value thresholding algorithm for matrix completion', *SIAM Journal on Optimization* **20**(4), 1956–1982.
URL: *http://dx.doi.org/10.1137/080738970*

Candès, E. J. & Tao, T. (2009), 'The power of convex relaxation: near-optimal matrix completion', *IEEE Transactions on Information Theory* **56(5)**, 2053–2080.

Candes, E. J., Wakin, M. B. & Boyd, S. (2008), 'Enhancing sparsity by reweighted l1 minimization', *Journal of Fourier Analysis and Applications* **14**(5), 877–905.

Candès, E. & Recht, B. (2008), 'Exact matrix completion via convex optimization', *Foundations of Computational Mathematics* **9**, 717–772.

Chen, S. & Donoho, D. (1994), On basis pursuit, Technical report, Department of Statistics Stanford University.

Cox, D. & Wermuth, N. (1996), *Multivariate Dependencies*, Chapman and Hall, London.

DeCoste, D. (2006), Collaborative prediction using ensembles of maximum margin matrix factorizations, *in* 'Proceedings of the 23rd International Conference on Machine Learning', ACM, pp. 249–256.

Dempster, A., Laird, N. & Rubin, D. (1977), 'Maximum likelihood from incomplete data via the EM algorithm (with discussion)', *Journal of the Royal Statistical Society Series B* **39**, 1–38.

Donoho, D. (2006), 'For most large underdetermined systems of equations, the minimal ℓ^1-norm solution is the sparsest solution', *Communications on Pure and Applied Mathematics* **59**, 797–829.

Donoho, D., Johnstone, I., Kerkyachairan, G. & Picard, D. (1995), 'Wavelet shrinkage; asymptopia? (with discussion)', *Journal of the Royal Statistical Society: Series B* **57**, 201–337.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004), 'Least angle regression (with discussion)', *Annals of Statistics* **32**(2), 407–499.

Fan, J. & Li, R. (2001), 'Variable selection via nonconcave penalized likelihood and its oracle properties', *Journal of the American Statistical Association* **96**(456), 1348–1360(13).

Fazel, M. (2002), Matrix Rank Minimization with Applications, PhD thesis, Stanford University.

Frank, I. & Friedman, J. (1993), 'A statistical view of some chemometrics regression tools (with discussion)', *Technometrics* **35**(2), 109–148.

Friedman, J. (2008), Fast sparse regression and classification, Technical report, Department of Statistics, Stanford University.

Friedman, J., Hastie, T., Hoefling, H. & Tibshirani, R. (2007), 'Pathwise coordinate optimization', *Annals of Applied Statistics* **2**(1), 302–332.

Friedman, J., Hastie, T. & Tibshirani, R. (2007), 'Sparse inverse covariance estimation with the graphical lasso', *Biostatistics* **9**, 432–441.

Friedman, J., Hastie, T. & Tibshirani, R. (2009), *glmnet: Lasso and elastic-net regularized generalized linear models.* R package version 1.1-4.
URL: *http://www-stat.stanford.edu/ hastie/Papers/glmnet.pdf*

Gao, H.-Y. & Bruce, A. G. (1997), 'Waveshrink with firm shrinkage', *Statistica Sinica* **7**, 855–874.

Gazit, H. (1991), 'An optimal randomized parallel algorithm for finding connected components in a graph', *SIAM J. on Computing* **20**(6), 1046–1067.

Grant, M. & Boyd., S. (2009), 'CVX: Matlab software for disciplined convex programming'. Web page and software available at `http://stanford.edu/~boyd/cvx`.

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction (Springer Series in Statistics)*, 2 edn, Springer New York.
URL: *"http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0387848576"*

Huang, J., Ma, S. & Zhan, C.-H. (2008), 'Adaptive lasso for sparse high-dimensional regression models', *Statistica Sinica* **18**, 1603–1618.

Ji, S. & Ye, J. (2009), An accelerated gradient method for trace norm minimization, *in* 'Proceedings of the 26th International Conference on Machine Learning', pp. 457–464.

Keshavan, R. H., Oh, S. & Montanari, A. (2009), 'Matrix completion from a few entries', *IEEE Transactions on Information Theory* **56(6)**, 2980–2998.

Knight, K. & Fu, W. (2000), 'Asymptotics for lasso-type estimators', *Annals of Statistics* **28**(5), 1356–1378.

Larsen, R. (2004), 'Propack-software for large and sparse svd calculations'. Available at `http://sun.stanford.edu/∼rmunk/PROPACK`.

Larsen, R. M. (1998), Lanczos bidiagonalization with partial reorthogonalization, Technical Report DAIMI PB-357, Department of Computer Science, Aarhus University.

Lauritzen, S. (1996), *Graphical Models*, Oxford University Press.

Liu, J., Ji, S. & Ye, J. (2009), *SLEP: Sparse Learning with Efficient Projections*, Arizona State University. Available at `http://www.public.asu.edu/∼jye02/Software/SLEP`.

Liu, Z. & Vandenberghe, L. (2009), 'Interior-point method for nuclear norm approximation with application to system identfication', *SIAM Journal on Matrix Analysis and Applications* **31(3)**, 1235–1256.

Lu, Z. (2009), 'Smooth optimization approach for sparse covariance selection', *SIAM J. on Optimization* **19**, 1807–1827.
URL: *http://portal.acm.org/citation.cfm?id=1654243.1654257*

Lu, Z. (2010), 'Adaptive first-order methods for general sparse inverse covariance selection', *SIAM J. Matrix Anal. Appl.* **31**, 2000–2016.
URL: *http://dx.doi.org/10.1137/080742531*

Ma, S., Goldfarb, D. & Chen, L. (2011), 'Fixed point and bregman iterative methods for matrix rank minimization', *Mathematical Programmming* **128**(1-2), 321–353.
URL: *http://dx.doi.org/10.1007/s10107-009-0306-5*

Mazumder, R., Friedman, J. & Hastie, T. (2011), 'Sparsenet: coordinate descent with non-convex penalties', *Journal of the American Statistical Association (Theory and*

Methods) **106(495)**, 1125–1138.
URL: *doi:10.1198/jasa.2011.tm09738*

Mazumder, R. & Hastie, T. (2011), 'The graphical lasso : New insights and alternatives', *available at arXiv:1111.5479v2* .
URL: *http://arxiv.org/abs/1111.5479v2*

Mazumder, R. & Hastie, T. (2012), 'Exact covariance thresholding into connected components for large-scale graphical lasso', *Journal of Machine Learning Research* **13**, 781794.
URL: *http://arxiv.org/abs/1108.3829*

Mazumder, R., Hastie, T. & Tibshirani, R. (2010), 'Spectral regularization algorithms for learning large incomplete matrices', *Journal of Machine Learning Research* **11**, 2287–2322.
URL: *http://jmlr.csail.mit.edu/papers/volume11/mazumder10a/mazumder10a.pdf*

Meinshausen, N. & Bühlmann, P. (2006), 'High-dimensional graphs and variable selection with the lasso', *Annals of Statistics* **34**, 1436–1462.

Nesterov, Y. (2003), 'Introductory lectures on convex optimization: Basic course', *Kluwer, Boston* .

Nesterov, Y. (2005), 'Smooth minimization of non-smooth functions', *Math. Program., Serie A* **103**, 127–152.

Nesterov, Y. (2007), Gradient methods for minimizing composite objective function, Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain. Tech. Rep, 76.

Nikolova, M. (2000), 'Local strong homogeneity of a regularized estimator', *SIAM J. Appl. Math.* **61**, 633–658.

Osborne, M., Presnell, B. & Turlach, B. (2000), 'A new approach to variable selection in least squares problems', *IMA Journal of Numerical Analysis* **20**, 389–404.

Recht, B., Fazel, M. & Parrilo, P. A. (2007), 'Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization'. Available at `http://www.citebase.org/abstract?id=oai:arXiv.org:0706.4138`.

Rennie, J. & Srebro, N. (2005), Fast maximum margin matrix factorization for collaborative prediction, *in* 'Proceedings of the 22nd International Conference on Machine Learning', ACM, pp. 713–719.

Rothman, A., Bickel, P., Levina, E. & Zhu, J. (2008), 'Sparse permutation invariant covariance estimation', *Electronic Journal of Statistics* **2**, 494–515.

Salakhutdinov, R., Mnih, A. & Hinton, G. E. (2007), Restricted Boltzmann machines for collaborative filtering, *in* 'Proceedings of the 24th International Conference on Machine Learning', AAAI Press, pp. 791–798.

Scheinberg, K., Ma, S. & Goldfarb, D. (2010), 'Sparse inverse covariance selection via alternating linearization methods', *NIPS* pp. 1–9.
URL: *http://arxiv.org/abs/1011.0097*

She, Y. (2009), 'Thresholding-based iterative selection procedures for model selection and shrinkage', *Electronic Journal of Statistics* .

SIGKDD, A. & Netflix (2007), Soft modelling by latent variables: the nonlinear iterative partial least squares (NIPALS) approach, *in* 'Proceedings of KDD Cup and Workshop'. Available at `http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html`.

Srebro, N., Alon, N. & Jaakkola, T. (2005), Generalization error bounds for collaborative prediction with low-rank matrices, *in* 'Advances in Neural Information Processing Systems 17', MIT Press, pp. 5–27.

Srebro, N. & Jaakkola, T. (2003), Weighted low-rank approximations, *in* 'Proceedings of the 20th International Conference on Machine Learning', AAAI Press, pp. 720–727.

Srebro, N., Rennie, J. & Jaakkola, T. (2005), Maximum-margin matrix factorization, *in* 'Advances in Neural Information Processing Systems 17', MIT Press, pp. 1329–1336.

Stein, C. (1981), 'Estimation of the mean of a multivariate normal distribution', *Annals of Statistics* **9**, 1131–1151.

Takacs, G., Pilaszy, I., Nemeth, B. & Tikk, D. (2009), 'Scalable collaborative filtering approaches for large recommender systems', *Journal of Machine Learning Research* **10**, 623–656.

Tarjan, R. E. (1972), 'Depth-first search and linear graph algorithms', *SIAM Journal on Computing* **1(2)**, 146160.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society, Series B* **58**, 267–288.

Tibshirani, R. (2011), 'Regression shrinkage and selection via the lasso: a retrospective', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73**(3), 273–282.

URL: *http://dx.doi.org/10.1111/j.1467-9868.2011.00771.x*

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. & Altman, R. B. (2001), 'Missing value estimation methods for DNA microarrays', *Bioinformatics* **17**(6), 520–525.

Tseng, P. (2001), 'Convergence of a block coordinate descent method for nondifferentiable minimization', *Journal of Optimization Theory and Applications* **109**, 475–494.

Tseng, P. & Yun, S. (2009), 'A coordinate gradient descent method for nonsmooth separable minimization', *Mathematical Programming B* **117**, 387–423.

van de Vijver, M. J., He, Y. D., van't Veer, L. J., Dai, H., Hart, A. A., Voskuil, D. W., Schreiber, G. J., Peterse, J. L., Roberts, C., Marton, M. J., Parrish, M., Atsma, D., Witteveen, A., Glas, A., Delahaye, L., van der Velde, T., Bartelink, H., Rodenhuis, S., Rutgers, E. T., Friend, S. H. & Bernards, R. (2002), 'A gene-expression signature as a predictor of survival in breast cancer', *N. Engl. J. Med.* **347**, 1999–2009.

Witten, D. & Friedman, J. (2011), 'A fast screening rule for the graphical lasso.', *Journal of Computational and Graphical Statistics: In Press.* . Report dated 3-12-2011.

Witten, D., Friedman, J. & Simon, N. (2011), 'New insights and faster computations for the graphical lasso', *Journal of Computational and Graphical Statistics* **20(4)**, 892–900. (Reference draft dated 9/8/2011).

Yuan, M. & Lin, Y. (2007), 'Model selection and estimation in the gaussian graphical model', *Biometrika* **94**(1), 19–35.

Yuan, X. (2009), 'Alternating direction methods for sparse covariance selection', *Methods* (August), 1–12.
URL: *http://www.optimization-online.org/DB-FILE/2009/09/2390.pdf*

Zhang, C. H. (2010), 'Nearly unbiased variable selection under mnimax concave penalty', *Annals of Statistics* **38(2)**, 894–942.

Zhang, C.-H. & Huang, J. (2008), 'The sparsity and bias of the lasso selection in high-dimensional linear regression', *Annals of Statistics* **36**(4), 1567–1594.

Zhang, T. (2009), Multi-stage convex relaxation for non-convex optimization, Technical report, Rutgers Tech Report.

Zhao, P. & Yu, B. (2006), 'On model selection consistency of lasso', *Journal of Machine Learning Research* **7**, 2541–2563.

Zhou, S., Rütimann, P., Xu, M. & Bühlmann, P. (2011), 'High-dimensional covariance estimation based on gaussian graphical models', *Journal of Machine Learning Research* **999888**, 2975–3026.
URL: *http://dl.acm.org/citation.cfm?id=2078183.2078201*

Zou, H. (2006), 'The adaptive lasso and its oracle properties', *Journal of the American Statistical Association* **101**, 1418–1429.

Zou, H., Hastie, T. & Tibshirani, R. (2007), 'On the degrees of freedom of the lasso', *Annals of Statistics* **35**(5), 2173–2192.

Zou, H. & Li, R. (2008), 'One-step sparse estimates in nonconcave penalized likelihood models', *The Annals of Statistics* **36**(4), 1509–1533.