Semismooth Newton Augmented Lagrangian Method for Solving Lasso Problems with Implementation in R

MyungWon Lee (s1687781)
Michael Renfrew (s1406083)

Year 4 Project
School of Mathematics
University of Edinburgh
11th March 2022

Abstract

Lasso (Least Absolute Shrinkage and Selection Operator) is a popular and widely used regression method that does linear regression and variable selection in the field of statistics and machine learning. In this paper, we aim to present an inexact Semismooth Newton augmented Lagrangian method to solve Lasso problems. With the advantage of the second order sparsity of the problem, we can reduce the expensive computational cost and yield a fast and highly efficient algorithm. In addition, we implement this algorithm in an ***R*** *package, apply it to real world data and validate its performance by conducting experiments against its competitors. We also apply our approach to genome-wide methylation profiling on human ageing rates.*

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*MyungWon Lee (s1687781)*
Michael Renfrew (s1406083))

This project report is submitted in partial fulfilment of the requirements for the degree of *BSc(Hons) Mathematics and Statistics & MMath(Hons) Mathematics*.

...To our loved ones...

Acknowledgements

We wish to express appreciation to our project supervisor, Dr. Daniel Paulin, for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times in our research and writing of this thesis. Through weekly project meetings, online and on-campus, he showed his enthusiasm for our research and its betterment. His immense knowledge, profound experience and professional expertise in convex optimisation, optimisation algorithms and statistics have enabled us to complete this project successfully with a strong theoretical basis.

This acknowledgement would remain incomplete without also expressing gratitude towards our personal tutors, Dr. Toby Bailey and Dr. Joan Simon Soler of their support and advice. We thank the other staff members of the School of Mathematics for providing sufficient resources and workshops to complete this research too.

Last but not least, our final year project is dedicated to our loved ones, Emily, and our parents for their unconditional love and support during the journey of our undergraduate studies.

Contents

Chapter 1

Introduction

The problem of linear regression in general seeks to find parameter estimates $\hat{\beta}$ that minimise the sum of squares given the observed data x, y, and β a vector:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \beta x_i)^2$$

In the case of massively underdetermined models, that is: those in which the length of the parameter vector m is much greater than the number of data points n, we will often want to solve an additional problem, which is to remove or set to zero as many parameter estimates as possible while retaining predictive accuracy. This will reveal which parameters have a tangible effect, and improves the explainability of the model. With ever-pressing emphasis on machine learning techniques in the era of big data, feature selection methods have become a research interest in the field of statistics and machine learning. Ranging from Lasso (Tibshirani 1996) [42] and Smoothly Clipped Absolute Deviation Penalty (SCAD) (Fan et al. 2001) [14], to the recent proposal on constrained Lasso by (Gaines et al. 2018) [19], many researchers have aimed to prove its effectiveness in different settings. The Lasso minimisation is formulated as

$$\min_{x} \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1 \tag{1.1}$$

where $A \in \mathbb{R}^{m \times n}$ is the design matrix (data), $b \in \mathbb{R}^m$ is the response variable vector, $x \in \mathbb{R}^n$ is the explanatory variable vector and $\lambda > 0$ is the regularisation parameter. We will focus on the Lasso proposed by (Tibshirani 1996) [42] which involves a constraint on a suitable norm of the vector β, although this can be computationally costly. It is therefore useful to implement computationally efficient and fast algorithms. Many optimisation algorithms have been proposed to tackle large-scale optimisation problems. For the first order approaches, there are Alternating direction method of multipliers (ADMM) (Boyd et al. 2011) [7], Gradient Descent Method and Sub-gradient Method (Bertsekas 2015) [3], Fast Iterative Shrinkage-Thresholding Algorithm (Beck et al. 2009) [2] and many more to mention. In addition to that, second order methods are gaining more attention with research on matrix-free Interior Point Method (Gondzio 2012, Fountoulakis et al. 2014) [22, 16], Forward-Backward Splitting-Newton (Xiao et al. 2018) [47]

and Semismooth Newton Augmented Lagrangian method (Li et al. 2018) [30]. Our paper proposes an implementation of the Semismooth Newton Augmented Lagrangian (**SSNAL**) Method based on the study proposed by (Li et al. 2018) [30] in **R**. By using the dual problem method, this algorithm brings computational advantages, uses a smaller number of variables and several constraints. Then, **SSNAL** solves the dual problem of (1.1) where the second order sparsity property is fulled exploited.

The **SSNAL** algorithm not only works for Lasso. There are many related works done with the **SSNAL** method on different settings of Lasso problems and general convex composite optimisation problems. Semidefinite Programming Problem (Zhao et al. 2010) [50] & (Li et al. 2018) [29], Lasso (Li et al. 2018) [30], Fused Lasso (Li et al. 2018) [31], Clustered Lasso (Lin et al. 2019) [33], Group Lasso (Zhang et al. 2020) [49] Constrained Lasso (Deng et al. 2020) [12], Elastic Net (Boschi et al. 2020) [4], nonsmooth Optimisation (Zhou et al. 2021) [51] and many more. Moreover, there is ongoing research to solve Lasso problems with other algorithms than **SSNAL**. These includes sparse approximation with Interior Point Method (De Simone et al. 2022) [11] and improved *glmnet* for ℓ_1-regularised logistic regression (Yuan et al. 2012) [48].

Thus, our paper is greatly inspired by the research and the prior implementation in **Matlab** by the research team of (Li et al. 2018) [30]. We benchmarked and debugged against the **Matlab** implementation, following the original code structure of (Li et al. 2018) [30] as closely as possible, but making the relevant changes and optimisations necessary for an **R** environment to build a package in **R** with $\mathbf{C}^{++}$ to aid statisticians who desire to solve sparse regression and Lasso problems.

This paper comprises 6 main chapters with several subsections and is organised as follows: in the next chapter, we will present the basic concepts of lasso regression, convex optimisation, and the maximal monotone operator. We then propose the idea of the augmented Lagrangian function and the concept of convergence rates which are the key concepts in the **SSNAL** algorithm. Chapter 3 provides the algorithmic framework and its global and local stopping criteria by analysing its asymptotic superlinear convergence. An efficient, readily implementable stopping criterion is suggested to solve the inner subproblem of **SSNAL**. In Chapters 4 and 5, we perform experiments to validate our implementation by comparing our results against the original results and *glmnet* (Friedman et al. 2010) [18]. We then conclude our paper in the final chapter.

Chapter 2

Literature Review & Preliminaries

2.1 Lasso Regression

Lasso regression introduces an ℓ_1 regulariser term to constrain the ℓ_1 norm (sum of absolute values) of the parameter vector (Tibshirani 1996) [42]. This operates as a "budget" which forces factors with a negligible effect towards zero. In other words, by adding the ℓ_1 norm to the problem, we can obtain sparse solutions. The method was introduced as an improvement on *ridge regression*, which uses an ℓ_2 regulariser term (sum of squared parameter estimates). It is named after a type of lance due to its effect of "reining in" parameters, but may also be referred to by its acronym *least absolute shrinkage and selection operator*.

The Lasso problem may be formally specified in one of two ways. The first is by a simple constrained optimisation:

$$\arg\min_{\beta_0,\beta} \left\{ \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \right\}, \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq t \tag{2.1}$$

In this case the budget is t. We can rewrite the same problem in matrix form:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}\|Ax - b\|_2^2 \\ \text{s.t.} \quad & \|x\|_1 \leq t \end{aligned} \tag{2.2}$$

It may also be formulated and specified by its Lagrangian form:

$$\min_{x} \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1$$

where $A \in \mathbb{R}^{m \times n}$ is the design matrix (data), $b \in \mathbb{R}^m$ is the response variable vector, $x \in \mathbb{R}^n$ is the explanatory variable vector and $\lambda > 0$ is the regularisation parameter. In this case the budget is varied by changing λ. It is simpler to work with the Lagrangian form in practice, and we do so here.

The effect of changing the budget is illustrated in Figure 2.1 (Hastie et al. 2009) [24]. Note that s is t relativised to the value of $\|\beta\|_1$ in the unconstrained case. The optimal value of the budget is often obtained in practice by cross-validation

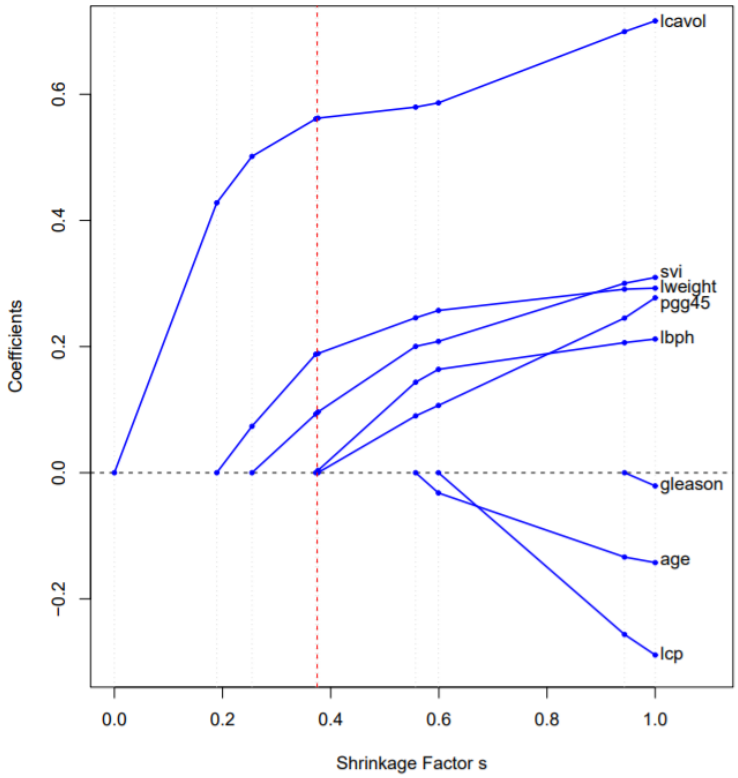


Figure 2.1: Effect of varying t

to maintain accuracy. Our package does not yet have functionality for this to be performed automatically, but for this example model the value chosen is indicated by the red line. Notice that only three parameters have been retained.

In order to see why Lasso regression in particular is inclined to set parameters to zero, consider the β_1 against β_2 plane for a 2-dimensional model in Figure 2.2 (Hastie et al. 2015) [25]. The blue diamond and circle are the constraint regions introduced by the budget for Lasso and ridge regression, respectively. Concentric ellipses represent solutions to the basic optimisation problem at progressively lower budgets. A solution is found at the intersection of the ellipse; the contours of equal residual sum of squares and the constraint region. This will tend to lie on axes for a diamond; that is, when one parameter is zero. In particular, if this is extended to higher dimensions, the diamond will take the form of a rhombus, with many vertices and flat edges allowing several parameters to be set to zero.

Finally, consider a generalisation of the problem in which we constrain not to a norm of β but to a certain number of non-zero estimates. This is known as *best-subset selection*, and while this is theoretically preferred to other approaches, it involves assessing each subset separately. This is however a combinatorial problem, so the computational complexity will grow as $O(m!)$. This is infeasible for most real-world applications. Although there are no closed-form solutions to Lasso problems, we can use various methods such as FISTA, ADMM and our proposed algorithm, **SSNAL**.

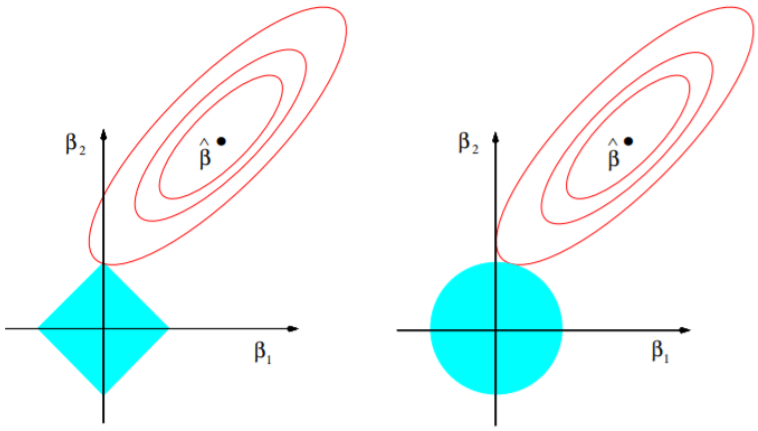


Figure 2.2: Ridge vs Lasso regression for two parameters

2.2 Convex Optimisation

In order to better understand the concepts of the paper, we review some fundamental elements of convex optimisation.

2.2.1 Subgradient

The first important concept is that of the *subgradient*, which is defined as follows (Hastie et al. 2015 & Rockafellar 2015) [25, 40]: the *subgradient* at a point x_0 is a number $c \in \mathbb{R}$ such that

$$f(x) - f(x_0) \geq c(x - x_0), \ \forall x \in \operatorname{dom} f \tag{2.3}$$

The *subdifferential* is a nonempty closed interval $[a, b]$ of permissible values for c and may be obtained by the left and right limit definitions:

$$\begin{aligned} a &= \lim_{x \to x_0^-} \frac{f(x) - f(x_0)}{x - x_0} \\ b &= \lim_{x \to x_0^+} \frac{f(x) - f(x_0)}{x - x_0} \end{aligned} \tag{2.4}$$

For instance, the function $y = |x|$ depicted in Figure 2.3 where y is not differentiable at $x = 0$, but admits several subgradients here. In particular, any line with a gradient between -1 and 1 inclusive will lie on or exclusively below $|x|$ as required. Therefore, the subdifferential here is the interval $[-1, 1]$. Note that we have differentiability at a point x_0 if and only if the subdifferential there is a singleton set. This idea can be generalised to vectors and dual spaces. If f is a convex vector function then any vector v such that $f(x) - f(x_0) \geq v \cdot (x - x_0)$ is a subgradient. The subdifferential is defined similarly and is denoted by $\partial f(x_0)$ $f(x) - f(x_0) \geq v^*(x - x_0)$, for a functional v^* in a dual space V^* defines our terms in this case.

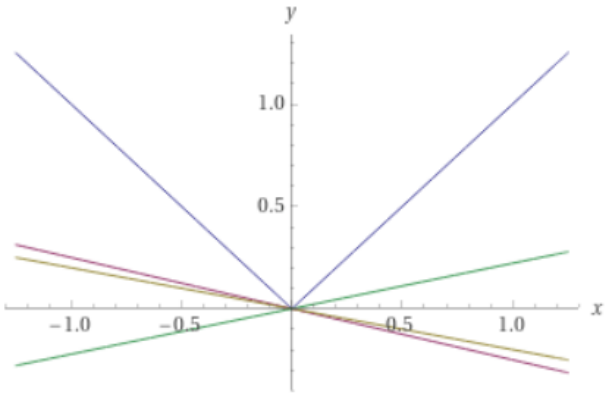


Figure 2.3: $y = |x|$ and other straight lines

2.2.2 Fenchel Conjugate

The Fenchel conjugate function (Fenchel 1949) [15], also known as a convex conjugate function, is defined as:

$$f^*(y) = \sup_{x \in \text{dom } f} (y^T x - f(x)) \tag{2.5}$$

where f^* is closed and convex (even when f is not). Then the following relations hold.

$$y \in \partial f(x) \iff x \in \partial f^*(y) \iff x^T y = f(x) + f^*(y) \tag{2.6}$$

As an example, the conjugate function of the Lasso penalty can be expressed as,

$$p^*(z) = \mathbb{I}_{\{\|z\|_\infty \leq \lambda\}} \begin{cases} 0, & \|z\|_\infty \leq \lambda \\ \infty, & \text{otherwise} \end{cases}$$

where p^* is an indicator function and a continuous differentiable function. In practice, there is a tight association between the convex conjugate and the solution to *Lagrangian* optimisation problem, which is discussed in next subsections.

2.2.3 Proximal Point Operators

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous convex function, (Rockafellar 1976a & Parikh et al. 2004) [38, 35] introduce the proximal point operator associated with p at x with $\sigma > 0$, where $\text{Prox}_p(\cdot)$ is defined as

$$\text{Prox}_p(x) := \arg\min_{u \in \mathcal{X}} \left\{ p(x) + \frac{1}{2}\|u - x\|^2 \right\}, \ \forall x \in \mathcal{X}$$

when p is differentiable and σ is sufficiently small, we have $\text{Prox}_{\sigma p}(x) \approx x - \nabla p(x)$. Moreover, by the Moreau identity, we have $x = \text{Prox}_{tp}(x) + t\,\text{Prox}\, t^{-1} p^*(x/t)$ for $t > 0$.

To implement **SSNAL**, we need the proximal mapping operator of the penalty function p and of its Fenchel Conjugate p^*. Thus, for each component $i = 1, \ldots, n$ of x, we have

$$\text{Prox}_{\sigma p}(x_i) = \begin{cases} x_i - \sigma\lambda, & x_i \geq \sigma\lambda \\ 0, & |x_i| < \sigma\lambda \\ x_i + \sigma\lambda, & x_i \leq -\sigma\lambda \end{cases} \tag{2.7}$$

$$\text{Prox}_{p^*/\sigma}(x_i/\sigma) = \begin{cases} \lambda, & x_i \geq \sigma\lambda \\ x_i/\sigma, & |x_i| < \sigma\lambda \\ -\lambda, & x_i \leq -\sigma\lambda \end{cases} \tag{2.8}$$

Then $\text{Prox}_{\sigma p}(x) = \sigma \,\text{Prox}_{p^*}(x/\sigma)$ holds.

2.2.4 Lagrangian Dual

Referring to (Boyd et al. 2004) [6], suppose we define an optimisation problem in a standard form:

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0, \ i = 1, \ldots, m, \\ & h_i(x) = 0, \ i = 1, \ldots, p. \end{aligned}$$

with variable $x \in \mathbb{R}^n$. We let the domain $\mathcal{D} = \bigcap_{i=0}^m \text{dom}\, f_i \cap \bigcap_{i=1}^p \text{dom}\, h_i$, which is nonempty, and its primal optimal value be p^*. In the general case, if we minimise $f_0(x)$ subject to $f_i(x) \leq 0$ and $h_i(x) = 0$, then we can define the *Lagrangian* as:

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x)$$

where the domain is defined as, $\text{dom}\, \mathcal{L} = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. λ_i and ν_i are the *Lagrangian multiplier*s associated with the ith inequality constraint $f_i(x) \leq 0$ and the ith equality constraint $h_i(x) = 0$ respectively. We then define the *Lagrangian dual function* as:

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} \left(f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x) \right)$$

In the particular case of our Lasso problem, we have only a single function f_i which must be less than or equal to zero, the ℓ_1 norm of the parameter vector β minus a suitable budget.

2.2.5 Weak & Strong Duality

We denote the optimal value of the *Lagrangian* dual problem as d^* and the best lower bound its optimal value is the optimal value of the primal problem denoted as p^*. If the inequality

$$d^* \leq p^*$$

holds, then we say that the *weak duality* property holds in the problem.

If the primal problem is convex and there exists an equality,

$$d^* = p^*$$

then we say that the *strong duality* property holds in the problem. In other words, $d^* - p^* = 0$ where the optimal duality gap is zero.

2.2.6 KKT Optimality Conditions

From (Boyd et al. 2004 & Gauraha 2018) [6, 20], *Karush-Kuhn-Tucker* (**KKT**) conditions state that for any optimisation problem with differentiable objective and constraint functions for which strong duality holds, any pair of primal and dual optimal points must satisfy the condition below,

- Stationary Condition : Subdifferntial (2.4) of $\mathcal{L}(x, \lambda, \nu)$ at (x^*, λ^*, ν^*)

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0$$

- Complementary Slackness

$$\lambda_i^* f_i(x^*) = 0, \ i = 1, \ldots, m$$

  The complementary slackness property establishes useful relations between the value of the ith primal variable and the slackness of the ith dual constraint. Hence, the property allows to us to construct a dual optimal solution by starting from a primal optimal solution.

- Primal Feasibility

$$f_i(x^*) \leq 0, \ i = 1, ..., m \text{ and } h_i(x^*) = 0, \ i = 1, ..., p$$

- Dual Feasibility

$$\lambda_i^* \geq 0, \ i = 1, \ldots, m$$

To sum up, we will use the condition as below to proceed to the augmented Lagrangian method that satisfies the **KKT** conditions.

$$\begin{aligned} f_i(x^*) &\leq 0, \ i = 1, ..., m \\ h_i(x^*) &= 0, \ i = 1, ..., p \\ \lambda_i^* &\geq 0, \ i = 1, ..., m \\ \lambda_i^* f_i(x^*) &= 0, \ i = 1, ..., m \\ \nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) &= 0 \end{aligned}$$

Here, f_i are convex functions and h_i are affine functions[1]. x^* minimises $\mathcal{L}(x, \lambda^*, \nu^*)$ over x, where its gradient vanishes at x^*.

2.3 Augmented Lagrangian Method

We begin by considering a simple optimisation problem on the addition of two functions f and h of x (Bouman, 2020) [5]:

$$\hat{x} = \arg\min_{x} \left(f(x) + h(x)\right)$$

Now in the case where f and h are not amenable to the same optimisation techniques, we may transform this into a bivariate optimisation problem in terms of x, v and constrain $x = v$, which appears to be a more difficult problem.

$$\hat{x} = \arg\min_{(x,v)} \left(f(x) + h(v)\right)$$

However we can add a penalty term so that we can remove the constraint.

$$\hat{x} = \arg\min_{(x,v)} \left(f(x) + h(v) + \frac{a}{2}\|x - v\|^2\right)$$

Now in the limit as $a \to \infty$, we obtain the solution to the original constrained problem. Sufficiently large a yields approximate solutions.

Finally, add a "pre-loading" u:

$$\hat{x} = \arg\min_{(x,v)} \left(f(x) + h(v) + \frac{a}{2}\|x - v + u\|^2\right)$$

The u which sets $x = v$ and optimises can then be found iteratively:

$$u_{n+1} = u_n + (x - v)$$

This can be thought of intuitively as converging approximately to the problem by displacing the $x - v$ component so that u dominates and $(x - v) \to 0$, i.e. $\to x = v$, and the constraint is restored.

The implementation of these general ideas in our particular semismooth Newton augmented Lagrangian (**SSNAL**) algorithm will become clear in the rest of the literature review and preliminaries.

2.4 Maximal Monotone Operator

We denote the maximal monotone operator of convex function f to be,

$$\mathcal{T}_f(x) := \partial f(x), \quad \mathcal{T}_f^{-1}(x') := \partial f^*(x')$$

[1] An affine function is a function composed of a sum of constant and a linear function given as $f(x) = A(x) + b$ for some matrix A and vector b of appropriate dimensions. i.e. a linear transformation followed by a translation.

We also denote the maximal monotone operator of convex-concave function l to be,

$$\mathcal{T}_l(y,z,x) := \{(y',z',-x') \in \partial l(y,z,x)\},$$
$$\mathcal{T}_l^{-1}(y',z',x') := \{(y,z,x)|(y',z',-x') \in \partial l(y,z,x)\}$$

now we need,
1. $\mathcal{T}_f$ to satisfy the error bound condition, the *metrically regular* condition for the origin with the constant a_f which is called the error bound modulus associated with $\mathcal{T}$. By (Cui et al. 2016) [10] and (Li et al. 2018) [30], when $\mathcal{T}^{-1}(0) \neq \emptyset, \exists \varepsilon > 0$ & $a > 0$ such that

$$\operatorname{dist}(\xi, \mathcal{T}(0)) \leq a\|\eta\|, \ \forall \eta \in \mathcal{B}(0,\varepsilon), \ \forall \xi \in \mathcal{T}^{-1}(\eta)$$

where $\operatorname{dist}(x,\mathcal{C}) := \inf_{x' \in \mathcal{C}} \|x - x'\|$ for any $x \in \mathcal{X}$ and any $\mathcal{C} \subset \mathcal{X}$ and $\mathcal{B}(0,\varepsilon) = \{y \in \mathcal{Y} : \|y\| < \varepsilon\}$ (open ball of radius ε about y) thus,

$$\operatorname{dist}(x, \mathcal{T}_f^{-1}(0)) \leq a_f \operatorname{dist}(0, \mathcal{T}_f(x)) \tag{2.9}$$

2. $\mathcal{T}_l$ is *metrically subregular* at (y^*,z^*,x^*) for the origin with the modulus a_l, by (Deng et al. 2020) [12], if there exist neighbourhoods $\mathcal{U}$ of (y^*,z^*,x^*) and $\mathcal{V}$ of 0 such that

$$\operatorname{dist}((y^*,z^*,x^*), \mathcal{T}_l^{-1}(0)) \leq a_l \operatorname{dist}(0, \mathcal{T}_l(y^*,z^*,x^*) \cap \mathcal{V}), \quad \forall (y^*,z^*,x^*) \in \mathcal{U} \tag{2.10}$$

Then, for any $r \in \operatorname{dom}(h)$, there exists a constant $\kappa_1 > 0$ and neighbourhood $\mathcal{E}_1$ of r such that $\forall r' \in \mathcal{E}$.

$$h(r') \geq h(r) + \langle \nabla h(r), r' - r\rangle + \kappa_1 \|r - r'\|^2 \tag{2.11}$$

Since $h(r)$ is 1-strongly convex, 2.11 holds with $\kappa_1 = 1$. ∂p is *metrically subregular* with constant $\kappa_2 > 0$, i.e. for any $(x,s) \in \operatorname{grp}(\partial p)$, there exists a constant $\kappa_2 > 0$ & a neighbourhood $\mathcal{E}_2$ of x such that for $\forall x' \in \mathcal{E}_2$

$$p(x') \geq p(x) + \langle s, x' - x\rangle + \kappa_2 \operatorname{dist}^2(x', (\partial p)^{-1}(s)) \tag{2.12}$$

$\partial(\|x\|_1)$ is *metric subregular* since the l_1 norm of a vector is a special case of nuclear norm $\|\cdot\|_*$ of a matrix.

2.5 Primal & Dual Problems

The primal function is defined as

$$(\mathbf{P}) \quad \max -\{f(x) := h(Ax) - \langle c, x\rangle + p(x)\} \tag{2.13}$$

where $A : \mathcal{X} \to \mathcal{Y}$, $h :\to \mathcal{Y} \to \mathbb{R}$, $p : \mathcal{X} \to (-\infty, +\infty]$ and $c \in \mathcal{X}$ is a vector. $\mathcal{X}$ and $\mathcal{Y}$ are two real finite dimensional Euclidean spaces each with an inner product

$\langle\cdot,\cdot\rangle$ and its induced norm $\|\cdot\|$.

The dual function is defined as

$$(\mathbf{D}) \quad \min\{h^*(\xi) + p^*(u) \mid A^*\xi + u = c\} \tag{2.14}$$

where, A^* is an adjoint matrix of A and $h^*(\cdot)$ and $p^*(\cdot)$ are Fenchel conjugate functions(2.5) of $h(\cdot)$ and $p(\cdot)$ respectively. We want to minimise the ℓ_1-regularised Lasso problem (1.1). Thus, we first set a mild assumption as below.

Assumption 1. $h^*(\cdot)$ is essentially smooth where h^* is differentiable on $\mathrm{int}(\mathrm{dom}\, h^*) \neq \emptyset$ and $\lim_{k\to\infty}\|h^*(\xi^k)\| = +\infty$ whenever $\{\xi^k\}$ is a sequence in $\mathrm{int}(\mathrm{dom}\, h^*)$ converging to a boundary point ξ of $\mathrm{int}(\mathrm{dom}\, h^*)$. $p^*(\cdot)$ is either an indicator function $\delta_p(\cdot)$ or a support function δ_p^* for some nonempty polyhedral convex set $P \subseteq \mathcal{X}$.

Then we define the primal function as

$$(\mathbf{P}) \quad \min\left\{\frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1\right\} \tag{2.15}$$

where $h(\xi) = \frac{1}{2}\|\xi - b\|^2$, $p(x) = \lambda\|x\|_1$, and $\mathrm{Prox}_{\sigma\lambda\|\cdot\|_1}(x) = \arg\min_u\{\frac{1}{2}\|u - x\|^2 + \sigma\lambda\|u\|_1\}$

The dual function is defined as (derivation in Appendix A.)

$$(\mathbf{D}) \quad \max\left\{-\frac{1}{2}\|\xi\|^2 + \langle b, \xi\rangle \mid A^T\xi + u - c = 0, \|u\|_\infty \leq \lambda\right\} \tag{2.16}$$

where $h^*(\xi) = -\frac{1}{2}\|\xi\|^2 + \langle b, \xi\rangle$, $p^*(x) = \mathbb{I}_{\|x\|_\infty}$ by (2.2.2) and $\mathrm{Prox}_{\sigma p}(x) = \mathrm{sgn}(x) \circ \max\{|x| - \sigma\lambda, 0\}$ by (2.7).

2.6 Augmented Lagrangian Function

Then the Lagrangian function of (**D**) is,

$$l(\xi, u; x) = h^*(\xi) + p^*(u) - \langle x, A^*\xi + u - c\rangle, \quad \forall(\xi, u; x) \in \mathcal{Y} \times \mathcal{X} \times \mathcal{X} \tag{2.17}$$

where $x \in \mathbb{R}^n$ is the *Lagrangian multiplier* and penalises the dual constraint's violation. By [**Proposition 12.60**] [43] and [**Corollary 4.4**] [21], we will further assume the following,

Assumption 2. $h : \mathcal{Y} \to \mathbb{R}$ is a convex differentiable function whose gradient is $\frac{1}{\alpha_h}$ *Lipschitz continuous* i.e.

$$\|\nabla h(\xi') - \nabla h(\xi)\| \leq \frac{1}{\alpha_h}\|\xi' - \xi\|, \quad \forall\xi', \xi \in \mathcal{Y} \tag{2.18}$$

($h^*(\cdot)$ is strongly convex with modulus α_k)

Assumption 3. h is essentially locally strongly convex, i.e. for any compact

and convex set $K \subset dom\ \partial h$, there exists $\beta_K > 0$ s.t.

$$(1-\lambda)h(\xi') + \lambda h(\xi) \leq h((1-\lambda)\xi' + \lambda\xi) + \frac{1}{2}\beta_K\lambda(1-\lambda)\|\xi' - \xi\|^2, \quad \forall \xi', \xi \in K \tag{2.19}$$

($\nabla h^*(\cdot)$ is locally *Lipschitz continuous* and directionally differentiable on $\mathrm{int}(\mathrm{dom}\, h^*)$)

Assumption 4. The Karush Kuhn Tucker (**KKT**) system is nonempty and its solution is denoted as $\bar{\xi}, \bar{u}$ and $\bar{x}$.

The **KKT** conditions suggested by (Boyd et al. 2004 & Gauraha 2018) [6, 20] in the context of Lasso are,

$$\begin{cases} 0 \in \delta h^*(\xi) - Ax, \\ 0 \in \partial p^*(u) - x, \\ 0 = A^*\xi + u - c, \\ (x, \xi, u) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{X} \end{cases}$$

As we assumed that **KKT** system has at least one solution and by **Assumption 1**, any points that satisfy the **KKT** conditions are primal and dual optimal and have zero duality gap by the Strong Duality Theorem 2.2.5. Thus, we can rewrite the **KKT** system as

$$\begin{cases} 0 \in \partial h^*(\xi) - Ax, \\ 0 \in \partial p^*(u) - x, \\ 0 = A^*\xi + u - c, \\ (x, \xi, u) \in \mathcal{X} \times \mathrm{int}(\mathrm{dom}\, h^*) \times \mathcal{X} \end{cases} \tag{2.20}$$

Therefore, the stationary condition, Slater's condition, primal and dual feasibility are shown above. More importantly, for solving the dual problem (2.14), we do not have to consider the entire domain but to focus on $\mathrm{int}(\mathrm{dom}\, h^*) \times \mathcal{X}$. Now we denote $(\bar{\xi}, \bar{u})$ to be an optimal solution to the dual problem (2.14), we also denote $\mathcal{M}(\bar{\xi}, \bar{u})$ as the set of *Lagrangian multipliers* associated with $(\bar{\xi}, \bar{u})$. Then with $\mathcal{M}(\bar{\xi}, \bar{u}) \neq \emptyset$ we can say that the second order sufficient condition (Rockafellar 2015) [40] for optimality of the dual problem (2.14) holds at $(\bar{\xi}, \bar{u})$ if h is twice differentiable, $\nabla h(\xi^*) = 0$ and $\nabla^2 h(\xi^*) \succ 0$ then ξ^* is a strict local minimum of h.

Following from **Assumption 4** and **[Theorem 1]** [30], the **KKT** system (2.20) has at least one solution and the second order sufficient condition for the dual problem holds at $(\bar{\xi}, \bar{u})$. $\mathcal{T}_l$ is metrically subregular (2.10) at $(\bar{\xi}, \bar{u}, \bar{x})$ for the origin. Then (Boyd et al. 2004) [6] suggests that $(\bar{\xi}, \bar{u}, \bar{x})$ solves the **KKT** (2.20) if and only if $(\bar{\xi}, \bar{u})$ and $(\bar{x})$ are the optimal solutions of (**D**) and (**P**), respectively. Given $\sigma > 0$, the augmented Lagrangian function of (**D**) is,

$$\mathcal{L}_\sigma(\bar{\xi}, \bar{u}; \bar{x}) := l(\xi, u; x) + \frac{\sigma}{2}\|A^*\xi + u - c\|^2, \ \forall (\xi, u; x) \in \mathcal{Y} \times \mathcal{X} \times \mathcal{X} \tag{2.21}$$

where the Lagrangian function $l(\xi, u; x)$ is defined in 2.17.

2.7 Convergence Rates

In this paper, we will introduce the superlinear convergence of the **SSNAL** algorithm, where it asymptotically converges Q-superlinearly and R-superlinearly. Referring to (Hintermüller, 2010) [26], we let $\{x^k\} \subset \mathbb{R}^n$ denote a sequence with limit $x^* \in \mathbb{R}^n$, and let $p \in [1, +\infty)$ then the quotient factor (Q-factor) is,

$$Q_p\{x^k\} := \begin{cases} \limsup_{k\to\infty} \frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|^p}, & \text{if } x^k \neq x^*,\ \forall k \geq k_0 \\ 0, & \text{if } x^k = x^*,\ \forall k \geq k_0 \text{ for some } k_0 \in \mathbb{N} \\ +\infty, & \text{otherwise} \end{cases}$$

and the quantity is called the Q-order of $\{x^k\}$ where,

$$O_Q\{x^k\} := \inf\{p \in [1, +\infty) : Q_p\{x_k\} = +\infty\}$$

There always exists a value $p_0 \in [1, +\infty)$ such that

$$Q_p\{x^k\} = \begin{cases} 0, & \text{for } p \in [1, p_0) \\ +\infty, & \text{for } p \in (p_0, +\infty) \end{cases}$$

As our interest lies in the linear convergence rate, the Q-factors depend on the norms used and the Q-order of 1 and 2 are suggested below.

$$\begin{cases} Q_1\{x^k\} = 0 : & \text{Q-superlinear convergence} \\ 0 < Q_1\{x^k\} < 1 : & \text{Q-linear convergence} \\ Q_2\{x^k\} = 0 : & \text{Q-superquadratic convergence} \\ 0 < Q_2\{x^k\} < 1 : & \text{Q-quadratic convergence} \end{cases} \tag{2.22}$$

Now we look at R-convergence. Let $\{x^k\} \subset \mathbb{R}$ denote a sequence with limit $x^* \in \mathbb{R}^n$ and let $p \in [1, +\infty)$ then the root convergence factor (R-factor) of $\{x^k\}$ is,

$$R_p\{x^k\} := \begin{cases} \limsup_{k\to\infty} \|x^k - x^*\|^{1/k}, & \text{if } p = 1 \\ \limsup_{k\to\infty} \|x^k - x^*\|^{1/p^k}, & \text{if } p > 1 \end{cases} \tag{2.23}$$

and the quantity is called the R-order of $\{x^k\}$ where,

$$O_R\{x^k\} := \inf\{p \in [1, +\infty) : R_p\{x_k\} = 1\}$$

In contrast with the Q-factor, the R-factor is independent of the norm and there always exists a value $p_0 \in [1, \infty)$ such that

$$R_p\{x^k\} = \begin{cases} 0, & \text{for } p \in [1, p_0) \\ 1, & \text{for } p \in (p_0, +\infty) \end{cases}$$

and the Q and R quantities are related as follows:

$$O_Q\{x^k\} \leq O_R\{x^k\} \quad \text{and} \quad R_1\{x^k\} \leq Q_1\{x^k\}$$

Chapter 3

Implementation of Algorithm

This section comprises two subsections. In the first subsection, we will introduce the inexact augmented Lagrangian method (**Algorithm 1**) to solve (**D**) (2.16) along with the description of global and local convergence criteria for the superlinear convergence of the algorithm under the mild assumption. In section 3.2, we will describe the Theorem of Semismoothness and semismooth Newton method (**Algorithm 2**) along with implementable stopping criteria.

3.1 Inexact Augmented Lagrangian Method

Algorithm 1 Inexact Augmented Lagrangian Method

Let $\sigma_0 > 0$ be a given penalty parameter. **Choose** $(\xi^0, u^0, x^0) \in \mathrm{int}(\mathrm{dom}\, h^*) \times \mathrm{dom}\, p^* \times \mathcal{X}$.

For $k = 0, 1, \ldots, \infty$, **perform** the following steps in each iteration,

(1) **Compute**

$$(\xi^{k+1}, u^{k+1}) \approx \arg\min\{\Psi_k(\xi, u) := \mathcal{L}_{\sigma_k}(\xi, u; x^k)\} \tag{3.1}$$

(2) **Compute**

$$x^{k+1} = x^k - \sigma_k(A^*\xi^{k+1} + u^{k+1} - c) \;\&\; \textbf{update} \quad \sigma_{k+1} \uparrow \sigma_\infty \leq \infty$$

We are interested in studying the convergence of the **SSNAL** algorithm. Here we define the term *local* to mean the fact that the convergence holds if x^0 is selected sufficiently close to x^* in a sequence $\{x^k\}$. For the term *global*, we define it as the convergence of a sequence $\{x^k\}$ to a local optimum at any given starting point.

In **Algorithm 1** (4), the subproblem (3.1) cannot be solved exactly on each iteration, however (Rockafellar 1976a, 1976b, 2015) [38, 39, 40] suggest a way to solve it. Rockafellar introduced a standard stopping criterion for the global convergence studied in [**Section 4**] [38], which the asymptotic Q-superlinear

convergence rate of $\{\xi^k\}$ is suggested under the *Lipschitz continuous* assumption of $\mathcal{T}^{-1}$ with respect to the origin.

3.1.1 Global Convergence

Given a nonnegative summable tolerance parameter sequence $\{\varepsilon_k\}$ and $\{\sigma_k\}$,

$$\Psi_k(\xi^{k+1}, u^{k+1}) - \inf \Psi_k \leq \frac{\varepsilon_k^2}{2\sigma_k}, \sum_{k=0}^{\infty} \varepsilon_k < \infty \tag{A}$$

Theorem 1. *Suppose that the solution set to (**P**) is nonempty. Then the sequence $\{x^k\}$ generated by the stopping criteria (A) is bounded and converges to an optimal solution of (**P**). In addition, $\{(\xi^k, u^k)\}$ is also bounded and converges to the unique optimal solution $(\bar{\xi}, \bar{u}) \in \operatorname{int}(\operatorname{dom} h^*) \times \operatorname{dom} p^*$ of (**D**).*

3.1.2 Local Convergence

Given a nonnegative summable tolerance parameter sequence $\{\delta_k\}$ and $\{\sigma_k\}$, the stopping criteria for local convergence defined by **[Section 4]** [38] is as below,

$$\Psi_k(\xi^{k+1}, u^{k+1}) - \inf \Psi_k \leq \left(\frac{\delta_k^2}{2\sigma_k}\right)\|x^{k+1} - x^k\|^2, \ \delta_k \geq 0, \ \sum_{k=0}^{\infty} \delta_k < \infty \tag{B1}$$

$$\operatorname{dist}(0, \partial\Psi_k(\xi^{k+1}, u^{k+1}) \leq \left(\frac{\delta_k'}{\sigma_k}\right)\|x^{k+1} - x^k\|, \ 0 \leq \delta_k' \to 0 \tag{B2}$$

Theorem 2. *Assume that the solution set Ω to (**P**) is nonempty. Suppose that **Assumption 2** and **Assumption 3** hold for $\mathcal{T}_f$ with modulus a_f (2.10). Then $\{x^k\}$ is convergent and for all k sufficiently large, the inequality below holds,*

$$\operatorname{dist}(x^{k+1}, \Omega) \leq \theta_k \operatorname{dist}(x^k, \Omega), \tag{3.2}$$

where $\theta_k = (a_f(a_f^2 + \sigma_k^2)^{-1/2} + 2\delta_k)(1 - \delta_k)^{-1} \to \theta_\infty = a_f(a_f^2 + \sigma_\infty^2)^{-1/2} < 1$ as $k \to +\infty$. In addition, if $\mathcal{T}_l$ is metrically subregular at $(\bar{\xi}, \bar{u}, \bar{x})$ for the origin with modulus a_l and the stopping criterion B2, then for all k sufficiently large,

$$\|(\xi^{k+1}, u^{k+1}) - (\bar{\xi}, \bar{u})\| \leq \theta_k' \|x^{k+1} - x^k\|$$

where $\theta_k' = a_l(1 + \delta_k')/\sigma_k$ with $\lim_{k\to\infty} \theta_k' = a_l/\sigma_\infty$. Moreover, the conclusions of Theorem 1 about $\{(\xi^k, u^k)\}$ are valid.

When $\sigma_\infty = +\infty$, the inequality (3.2) implies that $\{x^k\}$ converges with Q-superlinear convergence. (Cui et al. 2016) [10] suggest with sufficient conditions for ensuring the metric subregularity of the **KKT** conditions, dual feasibility and the dual objective function value converge asymptotically R-superlinearly as described in Section 2.7. Then by (**Proposition 4.1**) [10] we obtain the following theorem.

Theorem 3. *Let* $\{(\xi^k, u^k)\}$ *be a sequence generated under the condition B1. Then for all* k *sufficiently large such that* $\delta_k < 1$*, it holds that*

$$\begin{aligned} \|(\xi^{k+1}, u^{k+1}) - (\bar{\xi}, \bar{u})\| &\leq \theta'_k \|x^{k+1} - x^k\| \\ &\leq \theta'_k (1 - \delta_k)^{-1} \operatorname{dist}(x^k, \Omega) \end{aligned} \tag{3.3}$$

When $\sigma_\infty = +\infty$, the inequality (3.3) implies that $\{(\xi^k, u^k)\}$ converges with R-superlinear convergence. Hence, the **SSNAL** is guaranteed to produce a sequence that is asymptotically convergent to solve (**D**). Combing these 3 theorems leads to the asymptotic superlinear convergence of the augmented Lagrangian method to solve the Lasso problem.

3.2 Semismooth Newton Method

In this subsection, we aim to provide more detailed workings compared to (Li et al. 2018) [30]. With $h^*(\xi) = -\frac{1}{2}\|x\|^2 + \langle b, \xi\rangle$, $p^*(x) = \mathbb{I}_{\|x\|_\infty}$ and $\operatorname{Prox}_{\sigma p}(x) = \operatorname{sgn}(x) \circ \max\{|x| - \sigma\lambda, 0\}$ and from the level set,

$$\min_{\xi,u} \Psi(\xi, u) := \mathcal{L}_\sigma(\xi, u, \bar{x})$$

where $\sigma > 0$ and $\bar{x} \in \mathcal{X}$ we proceed to solve the inner subproblems in the augmented Lagrangian method (3.1).

3.2.1 Theorem of Semismoothness

Referring to (Mifflin 1977, Qi et al. 1993 & Schmidt 2010) [34, 37, 41], let us define the Theorem of Semismoothness. Suppose that $M : \mathcal{X} \rightrightarrows \mathcal{L}(\mathcal{X}, \mathcal{Y})$[1] & $F : \mathcal{X} \to \mathcal{Y}$, is a locally *Lipschitz continuous* function. F is said to be semismooth at $x \in \mathcal{X}$ if F is directionally differentiable at x and for any $G \in \partial F(x + \Delta x) = M(x + \Delta x)$ & $\Delta x \to 0$.

$$F(x + \Delta x) - F(x) - G(\Delta x) = o(\|\Delta\|)$$

F is said to be strongly semismooth at $x \in \mathcal{X}$ if,

$$F(x + \Delta x) - F(x) - G(\Delta x) = O(\|\Delta\|^2)$$

Then F is said to be semismooth (strongly semismooth) function on $\mathcal{X}$ if it is semismooth (strongly semismooth) everywhere in $\mathcal{X}$

(Qi et al, 2005) [36] and (Li et al. 2018) [30] mentioned all twice continuous differentiable functions and piecewise linear functions are strongly semismooth everywhere. Since $h^*(\cdot)$ is twice differentiable and $\operatorname{Prox}_{\lambda\|x\|_1}(\cdot)$ is a *Lipschitz continuous* piecewise affine function, we conclude that they are strongly semismooth.

[1] Multivalued functions or multifunctions are functions whose values are sets instead of of points. Given such a multifunction, $F : \mathbb{R}^n \to \mathbb{R}^m$ then $\Gamma_F := \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m | y \in F(x)\}$.

We newly denote for $\xi \in \mathcal{Y}$,

$$\begin{aligned}\psi(\xi) &:= \inf_z \Psi(\xi, u) = \inf \mathcal{L}_\sigma(\xi, u; x) \\ &= h^*(\xi) + p^*(\mathrm{Prox}_{p^*/\sigma}(\bar{x}/\sigma - A^*\xi + c) \\ &+ \frac{1}{2\sigma}\|\mathrm{Prox}_{\sigma p}(\bar{x} - \sigma(A^*\xi - c))\|^2 - \frac{1}{2\sigma}\|\bar{x}\|^2\end{aligned} \tag{3.4}$$

Suppose $(\bar{\xi}, \bar{u}) = \arg\min \Psi(\xi, u)$ where $(\bar{\xi}, \bar{u}) \in \mathrm{int}(\mathrm{dom}\, h^*) \times \mathrm{dom}\, p^*$, then

$$\begin{cases}\bar{\xi} = \arg\min \psi(\xi) \\ \bar{u} = \mathrm{Prox}_{p^*/\sigma}(\bar{x}/\sigma - A^*\bar{\xi} + c).\end{cases} \tag{3.5}$$

To obtain the value of ξ, we perform the second order derivative such that

$$\begin{aligned}\nabla\psi(\xi) &= \nabla h^*(\xi) - A\,\mathrm{Prox}_{\sigma p}(\bar{x} - \sigma(A^*\xi - c)),\ \forall \xi \in \mathrm{int}(\mathrm{dom}\, h^*) \\ \partial^2\psi(\xi) &= \nabla^2 h^*(\xi) + \sigma A\,\mathrm{Prox}_{\sigma p}(\bar{x} - \sigma(A^*\xi - c))A^*\end{aligned}$$

Since h^* is a convex function which is locally *Lipschitz continuous* and note that $\psi(\cdot)$ is strongly convex and continuously differentiable on $\mathrm{int}(\mathrm{dom}\, h^*)$, we define the new and well defined operator for the second order derivative as below,

$$\begin{aligned}\nabla\psi(\xi) &= \nabla h^*(\xi) - A\,\mathrm{Prox}_{\sigma p}(\bar{x} - \sigma(A^*\xi - c)),\ \forall \xi \in \mathrm{int}(\mathrm{dom}\, h^*) \\ \hat{\partial}^2\psi(\xi) &= \nabla^2 h^*(\xi) + \sigma A\,\mathrm{Prox}_{\sigma p}(\bar{x} - \sigma(A^*\xi - c))A^*\end{aligned} \tag{3.6}$$

where $\partial^2 h^*(\xi)$ is the Clarke subdifferential (Clarke 1990) [9] of $\nabla h^*(\cdot)$ at ξ and $\partial\,\mathrm{Prox}_{\sigma p}(\bar{x} - \sigma(A^*\xi - c))$ is the Clarke subdifferential of the *Lipschitz continuous* mapping and Jacobian of $\mathrm{Prox}_{\sigma p}(\cdot)$ at $\bar{x} - \sigma(A^*\xi - c)$. [**Proposition 2.3.3.** & **Theorem 2.6.6**] [9] suggests,

$$\partial^2\psi(\xi)(d) \subseteq \hat{\partial}^2\psi(\xi)(d),\ \forall d \in \mathcal{Y}$$

where if every element in $\partial^2\psi(\xi)$ is positive definite, so is every element in $\hat{\partial}^2\psi(\xi)$ Then we define the Hessian of ψ at ξ,

$$V := H + \sigma AUA^*$$

with $H \in \partial^2 h^*(\xi)$ and $U \in \partial\,\mathrm{Prox}_{\sigma p}(\bar{x} - \sigma(A^*\xi - c))$ where for the case of our paper, $h(\xi) = \frac{1}{2}\|\xi\|^2 + b^T\xi,\ \nabla h^*(\xi) = \xi + b,\ \nabla^2 h^*(\xi) = \mathbb{I}_m$. $h^*(\cdot)$ is twice differentiable and $\mathrm{Prox}_{\lambda\|\cdot\|}$ is a piecewise linear function which are strongly semismooth. To obtain $\bar{\xi}$, we solve the nonsmooth equation,

$$\nabla\psi(\xi) = 0,\ \xi \in \mathrm{int}(\mathrm{dom}\, h^*) \tag{3.7}$$

3.2.2 Solving the Subproblem

Now we proceed to Semismooth Newton Algorithm to solve the inner problem. By [**Theorem 14**] [30] and [**Theorem 3.5**] [50] we state the following theorem.

Algorithm 2 Semismooth Newton Method

Given the hyperparameter, $\mu \in (0, \frac{1}{2}), \tilde{\eta} \in (0,1), \tau \in (0,1]$ & $\delta \in (0,1)$.
Choose $\xi^0 \in \mathrm{int}(\mathrm{dom}\, h^*)$ and **iterate** the following steps for $j = 0,\ 1,\ \ldots$

(1) **Choose** $H_j \in \partial^2(\nabla h^*)(\xi^j)$ & $U_j \in \partial \,\mathrm{Prox}_{\sigma p}(\tilde{x} - \sigma(A^*\xi^j - c))$. **Let** $V_j := H_j + \sigma A U_j A^*$

Solve the following Linear System exactly or by the conjugate gradient algorithm to find d^j

$$\begin{aligned} V_j d + \nabla\psi(\xi^j) = 0, \text{where } d = \Delta\xi \\ \text{s.t. } \|V_j d^j + \nabla\psi(\xi^j)\| \le \min(\tilde{\eta}, \|\nabla\psi(\xi^j)\|)^{1+\tau} \end{aligned}$$

(2) **Set** $\alpha = \delta^{m_j}$ where m_j is the first nonnegative integer m for which,

$$\xi^j + \delta^m d^j \in \mathrm{int}(\mathrm{dom}\, h^*) \text{ and } \psi(\xi^j + \delta^m d^j) \le \psi(\xi^j) + \mu\delta^m \langle \nabla\psi(\xi^j), d^j\rangle$$

(3) **Set** $\xi^{j+1} = \xi^j + \alpha_j d^j$

Theorem 4. *Let the sequence ξ^j be generated by* ***Algorithm 2*** *(6), then ξ^j converges to the unique optimal solution $\bar{\xi} \in \mathrm{int}(\mathrm{dom}\, h^*)$ and $\bar{u} = \bar{x} - \sigma(A^T\bar{\xi} - c)$ at least superlinearly, i.e.,*

$$\|\xi^{j+1} - \bar{\xi}\| = O(\|\xi^j - \bar{\xi}\|^{1+\tau}),\ \textit{for any } j \ge 0, V_j \in \partial^2\psi(\xi^j) \tag{3.8}$$

Thus, the rate of convergence for **Algorithm 2** (6), is of order $1+\tau$. If $\tau = 0$, then we obtain a locally asymptotic Q-superlinear convergence rate.

We now expand the derivation of the implementable stopping criteria for (A), (B1) and (B2) that was not shown in the original paper. We first notice that,

$$\begin{aligned} &\Psi_k(\xi^{k+1}, u^{k+1}) = \inf_u \Psi_k(\xi^{k+1}, u) = \psi_k(\xi^{k+1}) \\ &\inf \Psi_k = \inf_{\xi,u} \Psi_k(\xi, u) = \inf_\xi \inf_u \Psi_k(\xi, u) = \inf_\xi \psi_k(\xi) = \inf \psi_k \\ &\text{then, } \Psi_k(\xi^{k+1}, u^{k+1}) - \inf \Psi_k = \psi_k(\xi^{k+1}) - \inf \psi_k \end{aligned}$$

combining this and the strong convexity assumption of h^* in (2.18) and (2.19), we have

$$\begin{aligned} &\psi_k(\bar{\xi}) - \psi_k(\xi^{k+1}) \ge \frac{1}{\alpha_h}(\langle \nabla\psi_k(\xi^{k+1}), \bar{\xi} - \xi^{k+1}\rangle + \frac{1}{2}\|\bar{\xi} - \xi^{k+1}\|^2) \\ \iff\ &\psi_k(\xi^{k+1}) - \psi_k(\bar{\xi}) \le -\frac{1}{\alpha_h}(\langle \nabla\psi_k(\xi^{k+1}), \bar{\xi} - \xi^{k+1}\rangle + \frac{1}{2}\|\bar{\xi} - \xi^{k+1}\|^2) \\ &\qquad = -\frac{1}{2\alpha_h}\|\bar{\xi} - \xi^{k+1} + \nabla\psi_k(\xi^{k+1})\|^2 + \frac{1}{2\alpha_h}\|\nabla\psi_k(\xi^{k+1})\|^2 \\ \iff\ &\psi_k(\xi^{k+1}) - \inf \psi_k \le \frac{1}{2\alpha_h}\|\nabla\psi_k(\xi^{k+1})\|^2 \end{aligned} \tag{3.9}$$

where $(\nabla\psi_k(\xi^{k+1}), 0) \in \partial\Psi_k(\xi^{k+1}, u^{k+1})$. Then by 3.9 we can achieve the following criteria. For equation (A),

$$\begin{aligned} \frac{1}{2\alpha_h}\|\nabla\psi_k(\xi^{k+1})\|^2 &\le \frac{\varepsilon_k^2}{2\sigma_k} \\ \iff \|\nabla\psi_k(\xi^{k+1})\|^2 &\le \frac{\varepsilon_k^2}{\sigma_k/\alpha_h} \\ \iff \|\psi_k(\xi^{k+1})\| &\le \frac{\varepsilon_k}{\sqrt{\sigma_k/\alpha_h}} \end{aligned}$$

by **[Proposition 3.2]** [32] this can also be expressed as below.

$$\operatorname{dist}(0, \partial\Psi_k(\xi^{k+1}, u^{k+1})) \le \varepsilon_k / \max(1, \sqrt{\sigma_k/\alpha_h}), \text{ where } \sum_{k=0}^{\infty}\varepsilon_k < \infty$$

For equation (B1),

$$\begin{aligned} &\frac{1}{2\alpha_h}\|\nabla\psi_k(\xi^{k+1})\|^2 \le \frac{\delta_k^2}{2\sigma_k}\|x^{k+1}-x^k\|^2 \\ &\text{where } x^{k+1} := x^k + \sigma_k(A*\xi^{k+1} + z^{k+1} - c) \\ \iff &\|\nabla\psi_k(\xi^{k+1})\|^2 \le \alpha_h\sigma_k\delta_k^2\|A^*\xi^{k+1} + u^{k+1} - c\|^2 \\ \iff &\|\nabla\psi_k(\xi^{k+1})\| \le \sqrt{\alpha_h\sigma_k}\delta_k\|A^*\xi^{k+1} + u^{k+1} - c\| \end{aligned}$$

by **[Proposition 3.2]** [32] this is also equivalent to,

$$\begin{aligned} \iff &\operatorname{dist}(0, \nabla\psi_k(\xi^{k+1})) \le \frac{\delta_k}{\max(1, \sigma_k/\alpha_h)} \cdot \|x^{k+1}-x^k\| \\ \iff &\operatorname{dist}(0, \partial\Psi_k(\xi^{k+1}, u^{k+1})) \le \frac{\delta_k}{\max(1, \sigma_k/\alpha_h)} \cdot \|x^{k+1}-x^k\| \end{aligned}$$

For equation (B2),

$$\begin{aligned} &\operatorname{dist}(0, \partial\Psi_k(\xi^{k+1}, u^{k+1}) \le \left(\frac{\delta_k'}{\sigma_k}\right)\|x^{k+1}-x^k\| \\ \iff &\|\nabla\psi_k(\xi^{k+1})\| \le \delta'\|A^*\xi^{k+1} + u^{k+1} - c\| \end{aligned}$$

To sum up, we obtain the new stopping criteria for the approximate computation,

$$\|\psi_k(\xi^{k+1})\| \le \frac{\varepsilon_k}{\sqrt{\sigma_k/\alpha_h}} \tag{A'}$$

$$\|\nabla\psi_k(\xi^{k+1})\| \le \sqrt{\alpha_h\sigma_k}\delta_k\|A^*\xi^{k+1} + u^{k+1} - c\| \tag{B1'}$$

$$\|\nabla\psi_k(\xi^{k+1})\| \le \delta'\|A^*\xi^{k+1} + u^{k+1} - c\|,\ 0 \le \delta_k' \to 0 \tag{B2'}$$

where $\sum_{k=0}^{\infty}\varepsilon_k < \infty$ and $\sum_{k=0}^{\infty}\delta_k < \infty$ and $\|\nabla\psi_k(\xi^{k+1})\|$ is sufficiently small.

Now we want to solve the following equation in step 1 of *Algorithm 2*,

$$(H + \sigma AUA^*)d = -\nabla\psi(\xi) \tag{3.10}$$

where $H \in \partial^2 h^*(\xi)$ is a sparse matrix, A is a matrix with basis $\mathbb{R}^{n\times m}$ and $U \in \partial\,\mathrm{Prox}_{\sigma\lambda\|\cdot\|_1}(u)$ with $u := \tilde{x} = \sigma(A^*\xi - c)$. We now follow the settings in (Li et al. 2018) [30]. Since, H is a symmetric and positive definite matrix, we let L be a nonsingular matrix such that $H = LL^T$ and U is a diagonal and idempotent matrix. Since we know h^* is a strongly convex function and H is a symmetric positive definite matrix, V is also a positive definite matrix. Then the the subproblem (3.10) can be rewritten as,

$$(\mathbb{I}_m + \sigma(L^{-1}A)U(L^{-1}A)^T)(L^T d) = -L^T\nabla\psi(\xi)$$

For convenience, we simplify the equation and rewrite as

$$(\mathbb{I}_m + \sigma\mathcal{A}U\mathcal{A}^T)d = -\nabla\psi(\xi) \tag{3.11}$$

where $\mathcal{A} = (L^{-1}A)$ and the equation (3.11) is the Newton system associated with the standard Lasso problem (2.15). Since $U \in \mathbb{R}^{n\times n}$ is a diagonal matrix, $U = \mathrm{diag(u)}$, the cost of computing $\mathcal{A}U\mathcal{A}^T$ and $\mathcal{A}U\mathcal{A}^T d$ where the cost is $O(m^2 n)$ and $O(mn)$, respectively. To fully exploit the sparsity of U, (Li et al. 2018) [30] it is suggested to always choose u_i where the ith element is given by

$$\mathrm{u}_i = \begin{cases} 0, \text{ if } |x_i| \le \sigma\lambda, \\ 1, \text{ otherwise} \end{cases} \quad \text{where } x = \bar{x} - \sigma(A^*\xi - c),\ i = 1, ..., n.$$

We let $\mathcal{J} := \{j \mid |x_j| > \sigma\lambda,\ j = 1, ..., n\}$ and $|\mathcal{J}| = r$, the cardinality of $\mathcal{J}$. Then we have

$$\mathcal{A}U\mathcal{A}^T = (\mathcal{A}U)(\mathcal{A}U)^T = \mathcal{A}_{\mathcal{J}}\mathcal{A}_{\mathcal{J}}^T \tag{3.12}$$

where $\mathcal{A}_{\mathcal{J}}$ is the sub-matrix of $\mathcal{A}$ with those columns contained in $\mathcal{J}$ preserved. Then, the cost of computing decreases to $O(m^2 r)$ and $O(mr)$ for $\mathcal{A}U\mathcal{A}^T$ and $\mathcal{A}U\mathcal{A}^T d$ respectively (shown in Figure 3.1 [30]). For the case when $r << m$, instead of factorising an $m \times m$ matrix, we can invert a much smaller, $r \times r$, matrix by using the Sherman-Morrison-Wood formula (Van et al 1996) [44],

$$(\mathbb{I}_m + \sigma\mathcal{A}U\mathcal{A}^T)^{-1} = (\mathbb{I}_m + \sigma\mathcal{A}_{\mathcal{J}}\mathcal{A}_{\mathcal{J}}^T)^{-1} = \mathbb{I}_m - \mathcal{A}_{\mathcal{J}}(\sigma^{-1}\mathbb{I}_r + \mathcal{A}_{\mathcal{J}}^T\mathcal{A}_{\mathcal{J}})^{-1}\mathcal{A}_{\mathcal{J}}^T \tag{3.13}$$

Thus, the total computation costs for solving the Newton linear system (3.11) are significantly reduced from $O(m^2(m + r))$ to $O(r^2(m + r))$ respectively. However, in practice we have to decide when to solve the linear system by either Cholesky decomposition or computing the inverse directly. When the number of the nonzero components of $\mathrm{Prox}_{\sigma\lambda\|\cdot\|_1}(x)$ is large, we apply the conjugate gradient algorithm where σ is small and the current point is relatively further away from the optimal solution.

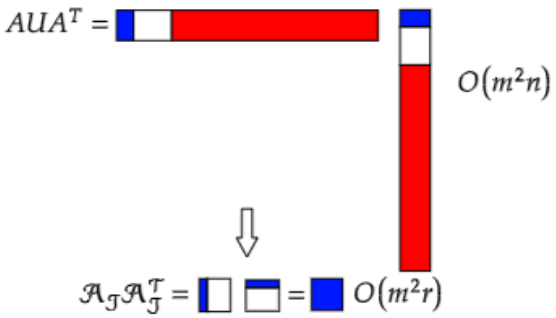


Figure 3.1: Reducing the computational costs from $O(m^2n)$ to $O(m^2r)$

Then, in step 2 in **Algorithm 2** (6), we perform the Line Search algorithm. By referring to the First Wolfe Condition (Wolfe 1969) [46],

$$f(\xi_j + \delta^m d^j) \leq f(\xi_j) - \delta^m \gamma,\ \gamma = \beta \nabla f(\xi_j)^T d^j > 0$$

where γ is proportional (factor of β) to the slope of the function along d^j at the current iterate x_j, then the general idea of the line search algorithm is that to minimise a function. In [**Theorem 3.5**] [50], d^j is always a descent direction, and we calculate the step along the descent direction, d

$$d^j = -H_j^{-1} \nabla f(\xi_j)$$

where $f : \mathbb{R}^n \to \mathbb{R},\ \xi_j \in \mathbb{R}^n,\ d^j \in \mathbb{R}^n$ and $\nabla f(\xi_j)^T d^j < 0$. Once we complete step 2, we proceed to step 3 to compute the sequence of $\{\xi^j\}$. In summary, we have demonstrated how the Semismooth Newton Algorithm can be implemented for solving Lasso regression problems.

Chapter 4

Experiments

In this chapter, we will illustrate our implementation with numerical experiments. According to what we discussed in Chapter 3, we now know that the dual approach is a more efficient choice since we will deal with sparse problems in this setting. We will specify our parameter settings on benchmark datasets and the methylation profiling dataset we used in the comparison with the original paper. When it comes to numeric calculation or analysis, it is undeniable that **Matlab** is a higher performing object oriented programming language compared to relatively slower language, **R**. However, our main focus is to assist mathematicians and statisticians to solve sparse problems in the domain of the **R** language. Thus, we expect our package developed in **R** with the aid of $\mathbf{C}^{++}$ to accomplish faster computation and boost time efficiency in large-scale dataset. For ease, we will now define our implementation as **R-SSNAL**.

4.1 Data

4.1.1 Benchmark Data

In this section, we conducted experiments on various datasets collected from the UCI data repository[1] [13] and Statlib[2] [45] in the LIBSVM format (Chang et al. 2011) [8]. Referring to (Huang et al. 2010) [27], we performed polynomial basis expansion to expand the original features. By utilising `genUCIdata.m` from the *SuiteLasso*[3] repository, *pyrim*, *triazines*, *abalone*, *bodyfat*, *housing*, *mpg* and *space_ga* were expanded in different orders but kept consistent with (Li et al. 2018) [30] so that we can have a parallel comparison. We also followed the naming convention where *abalone7* refers to a polynomial basis expansion of order 7 applied to expand the features of the original data *abalone*. m and n are number of samples and number of (expanded) features respectively and all the problem data are cases where $m << n$. The last column of Table 5.1 [30] refers to the largest eigenvalue of AA^* which is denoted as $\lambda_{max}(AA^*)$ and is the Lipschitz referred to in code as *Lip*. Through the number of dimensions and largest

[1] https://archive.ics.uci.edu/ml
[2] http://lib.stat.cmu.edu/datasets
[3] https://github.com/MatOpt/SuiteLasso

Data	Dimension(m; n)	$\lambda_{max}(AA^*)$
pyrim5	74; 201376	1.22e+06
triazines4	186; 635376	2.07e+07
abalone7	4177; 6435	5.21e+05
bodyfat7	252; 116280	5.29e+04
housing7	506; 77520	3.28e+05
mpg7	392; 3432	1.28e+04
space_ga9	3107; 5005	4.01e+03

Table 4.1: UCI and Statlib Testing Instances

eigenvalue, we can keep the data consistent while we are running the experiment.

4.1.2 Methylation Profiling Data

We further conducted the **R-SSNAL** algorithm on different datasets. Methylation Profiling Data was provided by (Hannum et al. 2013) [23] where it aimed to reveal quantitative views of human aging rates. The data consists of genome wide DNA methylation profiling of individuals across a large age range. The Illumina Infinium 450k Human DNA methylation Beadchip was used to obtain DNA methylation profiles across approximately 450k CpGs from human whole blood. With similar settings to the benchmark data, we performed cross-validation to determine the best regularising penalty, λ.

4.2 Parameter Tuning

We will follow the same setup as (Li et al. 2018) [30] for the aforementioned reason in the introduction. For the regularisation parameter, λ in the Lasso problem (1.1), we set $\lambda = \lambda_c \|A^*b\|_\infty$ where $\lambda_c \in (0, 1)$. In particular, we will use the values of λ_c as $1e-3$ and $1e-4$ to maintain consistency. We also measured the accuracy of an optimal solution $\bar{x}$ by using the following relative **KKT** residual:

$$\eta = \frac{\|\bar{x} - \mathrm{Prox}_{\lambda\|\cdot\|}(\bar{x} - A^*(A\bar{x} - b))\|}{1 + \|\bar{x}\| + \|A\bar{x} - b\|} < \varepsilon$$

For a given tolerance $\varepsilon > 0$, we will terminate the **R-SSNAL** algorithm when $\eta < \varepsilon$ and we set $\varepsilon = 10^{-6}$.

4.3 Objective Values

The objective value of the Lasso optimisation problem in **R-SSNAL** and the **Matlab** implementation of **SSNAL** was taken to be

$$\frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1 \tag{4.1}$$

This is the value returned in the primal objective object in the **SSNAL** implementations. In order to make appropriate comparisons with *glmnet* in **R**, which optimises on the objective:

$$\frac{1}{2}\frac{\|Ax-b\|_2^2}{m}+\lambda\|x\|_1 \tag{4.2}$$

It was necessary to suitably standardise and scale the response variables b and penalty factor λ on both inputs and outputs. In our tables, we will always refer to 4.1 when talking about the objective value. This was evaluated with the same function applied to output of *glmnet* and **R-SSNAL**

4.4 Number of Non-Zero Values (NNZ)

In order to estimate the number of non-zero values in the output (that is, parameter coefficients which are estimated to have an absolute value different from zero) from **R-SSNAL** and **Matlab** the formula

$$\text{nnz} := \min\left\{k \mid \sum_{t=1}^{k}|\hat{x}_t| \geq 0.999\|x\|_1\right\}$$

where the $\hat{x}_t$ are sorted in descending order, was employed, as in (Li et al. 2018)[30]. For *glmnet* results, the internal representation of NNZ was used.

4.5 Cross-Validation

A technique used commonly to determine how good a modelling procedure is is *n-fold cross validation*, in which a certain fraction of the data is left out (the training data) and used to make predictions on the remainder (test data) exactly n times. The mean-squared error (MSE), a standard measure in machine learning regression prediction analytics, is then calculated as $\sum(b_i-\hat{b}_i)^2/m$. Lower mean-squared errors are therefore better, representing a more accurate model.

4.6 Running Times

The computational results are obtained by running **R** (version 4.1.2) and **Matlab** (version 9.11) on a Windows 11 personal workstation (Quad-core, Intel(R) Core i5-10300H CPU @ 2.50GHz, 32G RAM). The running times for the **Matlab** implementation were taken from the *SuiteLasso* original code. For *glmnet* in **R**, and for **R-SSNAL**, the base **R**, `Rprof()`[4] profiler was used. All the computational experiments were performed using our personal laptops and faced relatively lower computational power compared to the original paper.

[4] https://github.com/r-prof/profile

Chapter 5

Comparison & Evaluation

This chapter details the results of running experiments with the three different packages implementing the **SSNAL** algorithm for the Lasso problem in **R** and **Matlab** across several different data sets: UCI and Statlib data, which is generally smaller and more instructive, and methylation data, which is much larger. We use as quantitative measures the primal objectives returned by the functions, as defined above, the running time, min/max parameter values and number of non-zeros. We also use n-fold cross validation to assess whether or not the **SSNAL** algorithms perform better than *glmnet* for these data.

5.1 UCI and Statlib data

5.1.1 R vs MATLAB

Firstly, in order to check that we had correctly implemented the algorithm as in *SuiteLasso* for **Matlab**, we checked that the minimum and maximum parameter estimates, as well as the number of non-zeros, were the same in both packages (Table 5.2).

For all seven data sets, at two levels of $\lambda_c = 10^{-3}, 10^{-4}$, the minimum and maximum values were almost identical for **Matlab** and our implementation in **R**. Cursory inspection of the sorted parameter vectors revealed that individual coefficients varied by at most around 1%, and preserved the sign.

For 13 simulations, the NNZ estimate from **Matlab** was the same as returned by **R**. It varied for $\lambda_c = 10^{-3}$ on the *triazines4* data set, but only by 7 non-zero values ($\sim$1%). It was noted that in general, with the same convergence stopping tolerance (*stoptol*) on η, and maximum number of iterations (*maxiter*), the **R-SSNAL** took more iterations to converge (on the order of an extra 20-40%).

We suspected that these very minor anomalies were probably attributable to numerical imprecision in **R**, different handling of floating point extrema, or other technical details, rather than a serious systemic bug, and were satisfied that we had well replicated the algorithm in **R**.

It is important to note that the algorithm ran much more slowly in R than in **Matlab** (Table 5.1), being on the order of 2-10 times slower for most data sets, but around 60 times slower for *triazines4* data with a relatively loose constraint

on $\|x\|_1$ of $\lambda_c = 10^{-4}$, which was computationally the most complex problem. Profiling the **R** code revealed that most of the time was spent on the linear system solution stage of the subproblem. Therefore, despite replicating closely the optimisations made by (Li et al. 2018) [30] in **Matlab**, such as preconditioned symmetric quasi-minimal residual[1] (PSQMR) (Freund 1997, Xin et al. 2018) [17, 28] and other preconditioning methods on the model matrix A, as well as using RcppEigen[2][3] (Bates et al. 2013) [1] based matrix-matrix and matrix-vector multiplications, we still suffered performance issues. Some of these will be ablated by compilation of the **R** code, but others will need careful attention if, as we intend, the package is to be released for general use. (The example code for the **R-SSNAL** method can be found in Appendix B.)

5.1.2 Primal Objective Values

As outlined above, the final objective values (Table 5.1) were calculated for the solution vectors returned by **R-SSNAL**, **Matlab**'s *SuiteLasso* and the industry-standard *glmnet*[4] in **R**, firstly to add another check that our **R-SSNAL** algorithm was correct but also to assess the performance of the algorithm. In the Lasso problem, we attempt to minimise the objective function, and therefore lower values represent "better" solutions for the given data.

For all data sets tried, at both values of λ_c, the objective value returned by our **R** code was the same as the **Matlab** implementation, as expected. The objective value for the **SSNAL** methods was however lower than that of the *glmnet* solution, around 5-10% lower for most data sets tried. Although the difference was not as stark for the *triazines* data at $\lambda_c = 10^{-3}$, it was particularly pronounced for the small *space_ga* data.

This discrepancy could have a number of causes. Firstly, the stopping tolerance for *glmnet* could be higher than our algorithm, and it may return before it has fully converged, or it could be that as *glmnet* principally employs first-order methods, our second-order Newtonian approach is simply better.

In practice, these small discrepancies may be industrially very relevant in such domains as the analysis of financial data, in which marginal improvements correspond to a large competitive edge.

5.1.3 Cross-Validation Accuracy

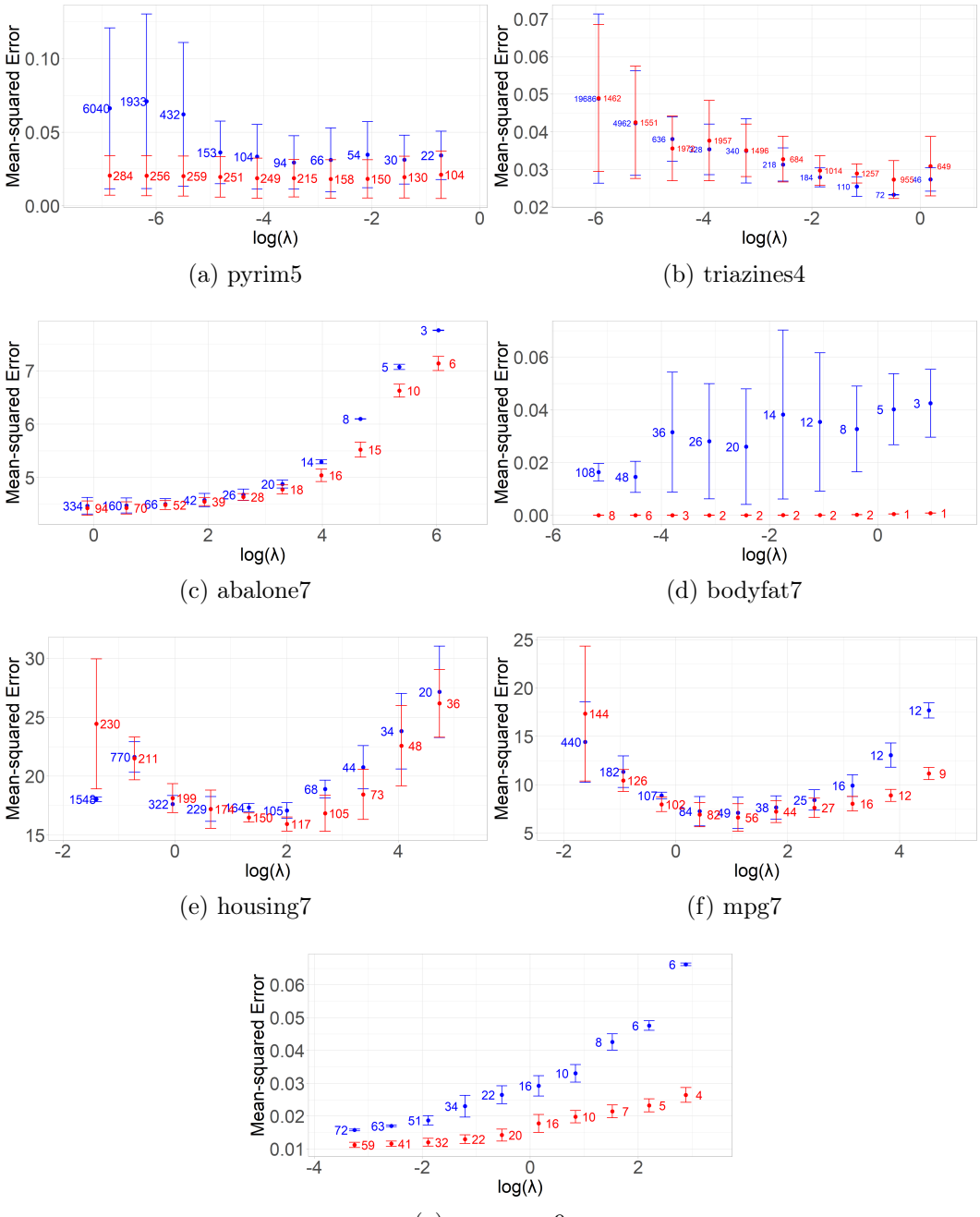
In order to assess whether the Lasso regression solutions returned by **R-SSNAL** outperformed those of *glmnet*, we performed a 2-fold cross-validation on the seven datasets of UCI and Statlib data at sensible values of the regularisation parameter λ close to their respective optima (Figure 5.1). The MSE returned by our package (using $\epsilon = 10^{-6}$ and $maxiter = 200$) was significantly lower than the *glmnet* estimates at most values of λ on many datasets, although it was particularly better for the *bodyfat* and *space_ga* data. Notice, importantly,

[1] https://github.com/cran/DWDLargeR
[2] https://github.com/RcppCore/RcppEigen
[3] https://stackoverflow.com/questions/35923787/fast-large-matrix-multiplication-in-r
[4] https://github.com/cran/glmnet

that these datasets also returned objective values several times lower at the two values of λ_c used to compare with the **Matlab** implementation in Table 5.1. We therefore expect that our improved primal objective values do in fact translate to meaningful improvements in learning and modelling accuracy. The NNZ figures (denoted by labels on the data points) were reasonably similar, and increased as expected as the regularisation parameter was diminished.

Data	λ_c	Time (s)	Objective Value
		Matlab | R-SSNAL | *glmnet*	Matlab | R-SSNAL | *glmnet*
pyrim5	10^{-3}	2.16 | 9.92 | 0.40	0.07511 | 0.07511 | 0.0795
(74;201376)		2.33 | 9.64 | 0.38	0.07511 | 0.07511 | 0.0795
	10^{-4}	2.63 | 27.82 | 0.26	0.0109 | 0.0108 | 0.0260
		2.53 | 26.96 | 0.24	0.0109 | 0.0108 | 0.0260
triazines4	10^{-3}	13.79 | 170.92 | 3.3	0.5452 | 0.5452 | 0.5548
(186;635376)		14.12 | 171.24 | 3.4	0.5452 | 0.5452 | 0.5548
	10^{-4}	27.90 | 1580.88 | 5.48	0.1156 | 0.1156 | 0.1524
		29.67 | 1477.62 | 5.10	0.1156 | 0.1156 | 0.1524
abalone7	10^{-3}	1.95 | 6.84 | 1.04	11407 | 11407 | 12158
(4177;6435)		2.17 | 6.82 | 1.04	11407 | 11407 | 12158
	10^{-4}	3.47 | 18.14 | 2.48	9289 | 9289 | 9716
		4.79 | 18.20 | 2.48	9289 | 9289 | 9716
bodyfat7	10^{-3}	1.64 | 5.00 | 0.30	0.2925 | 0.2925 | 1.334
(252;116280)		1.87 | 5.10 | 0.32	0.2925 | 0.2925 | 1.334
	10^{-4}	2.27 | 7.52 | 0.98	0.03031 | 0.03031 | 0.2372
		2.53 | 7.38 | 0.96	0.03031 | 0.03031 | 0.2372
housing7	10^{-3}	2.92 | 13.88 | 0.60	2775 | 2775 | 2819
(506;77520)		2.94 | 13.72 | 0.62	2775 | 2775 | 2819
	10^{-4}	2.27 | 7.52 | 0.98	920.3 | 920.3 | 987.1
		2.53 | 7.38 | 0.96	920.3 | 920.3 | 987.1
mpg7	10^{-3}	0.32 | 1.02 | 0.04	1669 | 1669 | 2076
(392;3432)		0.35 | 1.02 | 0.04	1669 | 1669 | 2076
	10^{-4}	0.37 | 2.90 | 0.16	890 | 890 | 985.57
		0.41 | 3.16 | 0.16	890 | 890 | 985.57
space_ga9	10^{-3}	0.81 | 2.34 | 0.10	31.9 | 31.9 | 62.08
(3107;5005)		0.94 | 2.40 | 0.12	31.9 | 31.9 | 62.08
	10^{-4}	2.27 | 7.52 | 0.98	19.88 | 19.88 | 31.63
		2.53 | 7.38 | 0.96	19.88 | 19.88 | 31.63

Table 5.1: Performance comparisons of SSNAL in Matlab and R and *glmnet*



(a) pyrim5 (b) triazines4

(c) abalone7 (d) bodyfat7

(e) housing7 (f) mpg7

(g) space_ga9

Figure 5.1: MSE comparison for R packages (Red: **R-SSNAL**, Blue: *glmnet*)

Data	λ_c	min	max	NNZ
		Matlab | R-SSNAL | *glmnet*	Matlab | R-SSNAL | *glmnet*	Matlab | R-SSNAL | *glmnet*
pyrim5	10^{-3}	-0.0422 | -0.0422 | -0.1732	0.165 | 0.166 | 0.1067	70 | 70 | 166
(74;201376)	10^{-4}	-0.0897 | -0.0896 | -0.63454	0.172 | 0.172 | 0.064	77 | 78 | 1643
triazines4	10^{-3}	-0.163 | -0.163 | -0.163	0.161 | 0.161 | 0.182	565 | 572 | 292
(186;635376)	10^{-4}	-0.458 | -0.458 | -0.4525	0.300 | 0.296 | 0.2465	261 | 261 | 1573
abalone7	10^{-3}	-8.13 | -8.13 | -13.49	11.7 | 11.7 | 11.7	24 | 24 | 21
(4177;6435)	10^{-4}	-13.3 | -13.3 | -12.97	16.1 | 16.1 | 7.96	59 | 59 | 129
bodyfat7	10^{-3}	-0.0465 | -0.0465 | -0.8133	1.05 | 1.05 | 1.202	2 | 2 | 17
(252;116280)	10^{-4}	-0.0526 | -0.0526 | -1.06	1.05 | 1.045 | 1.314	3 | 3 | 49
housing7	10^{-3}	-7.37 | -7.37 | -8.02	3.25 | 3.25 | 4.114	158 | 158 | 163
(506;77520)	10^{-4}	-13.1 | -13.1 | -19.7	11.3 | 11.27 | 8.39	281 | 281 | 484
mpg7	10^{-3}	-5.08 | -5.08 | -23.68	17 | 16.98 | 14.99	47 | 47 | 46
(392;3432)	10^{-4}	-11.8 | -11.8 | -18.93	15.3 | 15.3 | 16.38	128 | 128 | 172
space_ga9	10^{-3}	-1.14 | -1.14 | -3.68	0.978 | 0.978 | 3.77	14 | 14 | 19
(3107;5005)	10^{-4}	-3.56 | -3.56 | -4.59	2.64 | 2.64 | 4.71	38 | 38 | 58

Table 5.2: Performance comparisons of **SSNAL** in **Matlab** and **R** and *glmnet*

5.2 Methylation Data

We first performed a 10-fold cross-validation using `cv.glmnet()` to a data set consisting of 656 subjects' methylation β levels across approximately 450,000 loci in order to determine an appropriate range of λ with which to run the **R-SSNAL** implementation.

We were unable to run a 10-fold cross-validation on methylation data using **R-SSNAL** due to computational power and optimisation restraints which prohibited convergence in reasonable time with sensible *stoptol*, ϵ and *maxiter*, particularly at lower values of λ. This is to be expected since having a relatively looser constraint on the parameters would increase the number of non-zero values, and thus decrease the sparsity of the Hessian on which the efficiency of this algorithm depends.

We did however compare the objective values returned by **R-SSNAL** and *glmnet* at 11 values of λ around the optimimum returned by `cv.glmnet` (Figure 5.2) and found that in this setting the objective value returned by **R-SSNAL** was indeed around 2-3% lower than that returned by *glmnet*. It is therefore to be expected that if, as with the UCI and Statlib data, lower primal objectives translate to better MSEs in a cross-validation, this algorithm would do better on these data than *glmnet* with unrestrained computing power.

It is interesting to note that (Hannum et al. 2013) [23] attributed residual unexplained variance in the age prediction accuracy using the methylome to "over-" or "under-aging", and similar work has suggested clinical outcomes to be correlated to this residual. Although not directly comparable to that paper, which employed an *elastic net regression*, our findings show that at least some of the residual

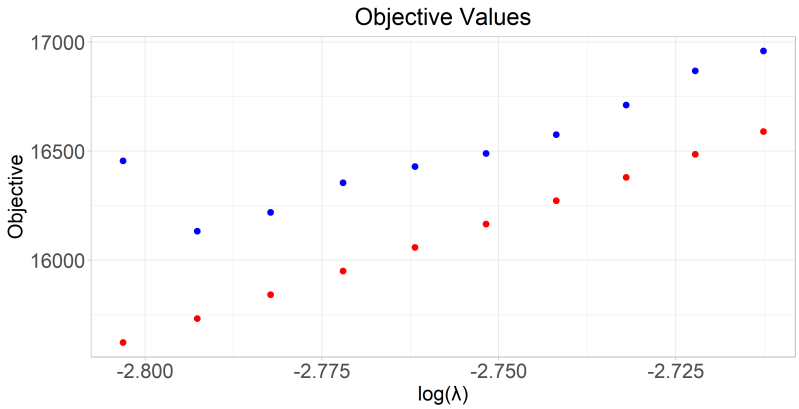


Figure 5.2: Objectives on methylation data for **R** packages (Red: **R-SSNAL**, Blue: *glmnet*)

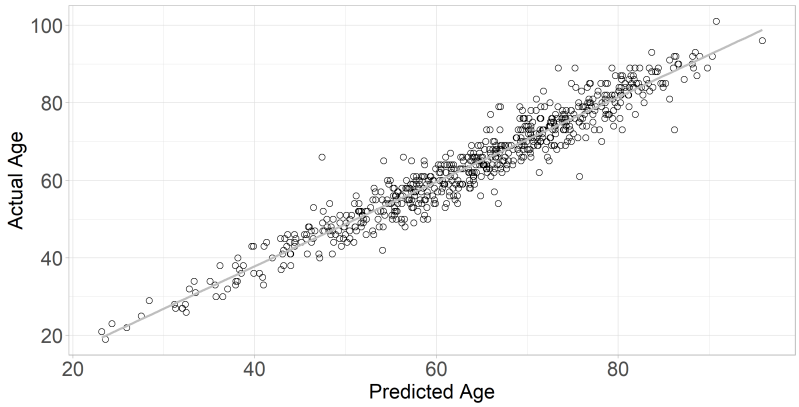


Figure 5.3: Cross-validation results at optimal λ

variance can be sometimes be attributed to improper convergence of the algorithm used, and better algorithms may do better at specifying this residual, if it exists.

The results of a 10-fold cross-validation at the optimal λ are shown for interest in Figure 5.3. Predicted age is very strongly linearly correlated with the actual age ($\sigma \approx 6.6$), so with these results, the methylome can probably be used to sensibly pinpoint the decade of age.

Chapter 6

Conclusion

In this paper, we have presented the fundamental principles underlying multivariate linear regression. In high-dimensional, underdetermined settings, such as those encountered in machine learning, or in data with a high degree of sparsity, it is useful to regularise the ℓ_1 norm of the coefficient vector β, such that a substantial number of regressors are set to zero, resulting in more robust, explainable models.

This Lasso technique results in a convex optimisation problem, with Lagrangian dual, for which several efficient first- and second-order algorithms have been designed. We replicate in the **R** language the work of (Li et al. 2018) [30], in which they propose and implement their semismooth Newton augmented Lagrangian algorithm in **Matlab**.

The ingenuity of the algorithm is predicated mostly on the sparsity of the Hessian matrix involved in solving the subproblem required for Newton's method - making this a second-order method. Ordinarily, square matrix inversion would be $\mathcal{O}(n^3)$, and so making this matrix smaller results in a massive speed-up, since most computation work is done here. Other optimisations are also made on it, such as quasi-minimal residual (PSQMR) decomposition and preconditioning.

Overall, however, the algorithm in base **Matlab** was slower than *glmnet* in **R**. We were able to replicate the results of the **Matlab** package, entitled *SuiteLasso*, in **R**, although it was slower than both. We intend to optimise further, compile the code, and - with the consent of the original authors - release an **R** package for general use.

We have demonstrated that our code in **R** works as intended and, with several datasets, have shown that it is capable of generating better objective values for the Lasso minimisation problem, which hold up in cross-validation by yielding better MSE values, than the most widely used package *glmnet*.

In future, we aim to explore **SSNAL** further, which can in theory be applied to any suitably specified optimisation problem, such as *elastic net regression* (Boschi et al. 2020) [4], which is a regularised regression that interpolates between the Lasso and ridge regressions, taking these as edge cases, according to the objective $\|Ax - b\|_2^2/2 + \alpha\lambda\|x\|_1 + (1-\alpha)\lambda\|x\|_2$. It may also be useful for the even more general setting of constraining the multivariate linear regression on the q-norm (for positive rational q) of the parameter vector. To that, we will finalise by publishing our package on CRAN.

Bibliography

[1] Bates, Douglas and Eddelbuettel, Dirk. "Fast and elegant numerical linear algebra using the RcppEigen package". In: *Journal of Statistical Software* 52 (2013), pp. 1–24.

[2] Beck, Amir and Teboulle, Marc. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[3] Bertsekas, Dimitri. *Convex optimization algorithms*. Athena Scientific, 2015.

[4] Boschi, Tobia, Reimherr, Matthew, and Chiaromonte, Francesca. "An Efficient Semi-smooth Newton Augmented Lagrangian Method for Elastic Net". In: *arXiv preprint arXiv:2006.03970* (2020).

[5] Bouman, Charles. *ECE641 - Lecture 22: Augmented Lagrangian for Constrained Optimization*. 2020.

[6] Boyd, Stephen, Boyd, Stephen P, and Vandenberghe, Lieven. *Convex optimization*. Cambridge university press, 2004.

[7] Boyd, Stephen, Parikh, Neal, and Chu, Eric. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[8] Chang, Chih-Chung and Lin, Chih-Jen. "LIBSVM: a library for support vector machines". In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), pp. 1–27.

[9] Clarke, Frank H. *Optimization and nonsmooth analysis*. SIAM, 1990.

[10] Cui, Ying, Sun, Defeng, and Toh, Kim-Chuan. "On the asymptotic superlinear convergence of the augmented Lagrangian method for semidefinite programming with multiple solutions". In: *arXiv preprint arXiv:1610.00875* (2016).

[11] De Simone, Valentina et al. "Sparse Approximations with Interior Point Methods". In: *Siam review* (2022).

[12] Deng, Zengde, Yue, Man-Chung, and So, Anthony Man-Cho. "An Efficient Augmented Lagrangian-Based Method for Linear Equality-Constrained Lasso". In: (2020), pp. 5760–5764.

[13] Dua, Dheeru and Graff, Casey. *UCI Machine Learning Repository*. 2017. URL: https://archive.ics.uci.edu/ml.

[14] Fan, Jianqing and Li, Runze. "Variable selection via nonconcave penalized likelihood and its oracle properties". In: *Journal of the American statistical Association* 96.456 (2001), pp. 1348–1360.

[15] Fenchel, Werner. "On conjugate convex functions". In: *Canadian Journal of Mathematics* 1.1 (1949), pp. 73–77.

[16] Fountoulakis, Kimon, Gondzio, Jacek, and Zhlobich, Pavel. "Matrix-free interior point method for compressed sensing problems". In: *Mathematical Programming Computation* 6.1 (2014), pp. 1–31.

[17] Freund, Roland W. "Preconditioning of symmetric, but highly indefinite linear systems". In: *15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*. Vol. 2. 1997, pp. 551–556.

[18] Friedman, Jerome, Hastie, Trevor, and Tibshirani, Rob. "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1 (2010), p. 1.

[19] Gaines, Brian R, Kim, Juhyun, and Zhou, Hua. "Algorithms for fitting the constrained lasso". In: *Journal of Computational and Graphical Statistics* 27.4 (2018), pp. 861–871.

[20] Gauraha, Niharika. "Introduction to the LASSO". In: *Resonance* 23.4 (2018), pp. 439–464.

[21] Goebel, Rafal and Rockafellar, R Tyrrell. "Local strong convexity and local Lipschitz continuity of the gradient of convex functions". In: *Journal of Convex Analysis* 15.2 (2008), p. 263.

[22] Gondzio, Jacek. "Matrix-free interior point method". In: *Computational Optimization and Applications* 51.2 (2012), pp. 457–480.

[23] Hannum, Gregory et al. "Genome-wide methylation profiles reveal quantitative views of human aging rates". In: *Molecular cell* 49.2 (2013), pp. 359–367.

[24] Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. *The elements of statistical learning - data mining, inference and prediction*. Springer, 2009.

[25] Hastie, Trevor, Tibshirani, Robert, and Wainwright, Martin. *Statistical learning with sparsity - the Lasso and generalizations*. CRC Press, 2015.

[26] Hintermüller, Michael. "Semismooth Newton methods and applications". In: *Department of Mathematics, Humboldt-University of Berlin* (2010).

[27] Huang, Ling et al. "Predicting execution time of computer programs using sparse polynomial regression". In: *Advances in neural information processing systems* 23 (2010), pp. 883–891.

[28] Lam, Xin Yee et al. "Fast algorithms for large-scale generalized distance weighted discrimination". In: *Journal of Computational and Graphical Statistics* 27.2 (2018), pp. 368–379.

[29] Li, Xudong, Sun, Defeng, and Toh, Kim-Chuan. "QSDPNAL: A two-phase augmented Lagrangian method for convex quadratic semidefinite programming". In: *Mathematical Programming Computation* 10.4 (2018), pp. 703–743.

[30] Li, Xudong, Sun, Defeng, and Toh, Kim-Chuan. "A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems". In: *SIAM Journal on Optimization* 28.1 (2018), pp. 433–458.

[31] Li, Xudong, Sun, Defeng, and Toh, Kim-Chuan. "On efficiently solving the subproblems of a level-set method for fused lasso problems". In: *SIAM Journal on Optimization* 28.2 (2018), pp. 1842–1866.

[32] Li, Xudong, Sun, Defeng, and Toh, Kim-Chuan. "An asymptotically superlinearly convergent semismooth Newton augmented Lagrangian method for Linear Programming". In: *SIAM Journal on Optimization* 30.3 (2020), pp. 2410–2440.

[33] Lin, Meixia et al. "Efficient sparse semismooth Newton methods for the clustered Lasso problem". In: *SIAM Journal on Optimization* 29.3 (2019), pp. 2026–2052.

[34] Mifflin, Robert. "Semismooth and semiconvex functions in constrained optimization". In: *SIAM Journal on Control and Optimization* 15.6 (1977), pp. 959–972.

[35] Parikh, Neal and Boyd, Stephen. "Proximal algorithms". In: *Foundations and Trends in optimization* 1.3 (2014), pp. 127–239.

[36] Qi, Liqun, Shapiro, Alexander, and Ling, Chen. "Differentiability and semismoothness properties of integral functions and their applications". In: *Mathematical Programming* 102.2 (2005), pp. 223–248.

[37] Qi, Liqun and Sun, Jie. "A nonsmooth version of Newton's method". In: *Mathematical programming* 58.1 (1993), pp. 353–367.

[38] Rockafellar, R Tyrrell. "Augmented Lagrangians and applications of the proximal point algorithm in convex programming". In: *Mathematics of operations research* 1.2 (1976), pp. 97–116.

[39] Rockafellar, R Tyrrell. "Monotone operators and the proximal point algorithm". In: *SIAM journal on control and optimization* 14.5 (1976), pp. 877–898.

[40] Rockafellar, Ralph Tyrell. *Convex analysis*. Princeton university press, 2015.

[41] Schmidt, Mark. "Graphical model structure learning with l1-regularization". In: *University of British Columbia* (2010).

[42] Tibshirani, Robert. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[43] Tyrrell Rockafellar, R and Wets, Roger J-B. "Variational analysis". In: *Grundlehren der mathematischen Wissenschaften* 317 (1998).

[44] Van Loan, Charles F and Golub, G. "Matrix computations (Johns Hopkins studies in mathematical sciences)". In: (1996).

[45] Vlachos, Panetelis. *StatLib*. 2005. URL: http://lib.stat.cmu.edu/datasets.

[46] Wolfe, Philip. "Convergence conditions for ascent methods". In: *SIAM review* 11.2 (1969), pp. 226–235.

[47] Xiao, Xiantao et al. "A regularized semi-smooth Newton method with projection steps for composite convex programs". In: *Journal of Scientific Computing* 76.1 (2018), pp. 364–389.

[48] Yuan, Guo-Xun, Ho, Chia-Hua, and Lin, Chih-Jen. "An improved glmnet for l1-regularized logistic regression". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 1999–2030.

[49] Zhang, Yangjing et al. "An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems". In: *Mathematical Programming* 179.1 (2020), pp. 223–263.

[50] Zhao, Xin-Yuan, Sun, Defeng, and Toh, Kim-Chuan. "A Newton-CG augmented Lagrangian method for semidefinite programming". In: *SIAM Journal on Optimization* 20.4 (2010), pp. 1737–1765.

[51] Zhou, Yuhao et al. "A semi-smooth Newton based augmented Lagrangian method for nonsmooth optimization on matrix manifolds". In: *arXiv preprint arXiv:2103.02855* (2021).

Appendix A

Proofs

A.1 Deriving Lagrangian Dual

We begin by letting $Ax = \tau \in \mathbb{R}^m$. Then the Lagrangian primal form (2.15) is equivalent to

$$D = \min_{x \in \mathbb{R}^n,\ \tau \in \mathbb{R}^m} \|\tau - b\|^2 + \lambda \|x\|_1 : \tau = Ax$$

Now we introduce another Lagrange multiplier, $\xi \in \mathbb{R}^m$

$$D = \min_{x \in \mathbb{R}^n,\ \tau \in \mathbb{R}^m} \max_{\xi \in \mathbb{R}^m} \|\tau - b\|^2 + \lambda \|x\|_1 + \langle \xi, Ax - \tau \rangle \tag{1}$$

By Slater's Condition, we can exchange the min and max in (1) to conclude

$$\begin{aligned} D &= \max_{\xi \in \mathbb{R}^m} \min_{x \in \mathbb{R}^n,\ \tau \mathbb{R}^m} \|\tau - b\|^2 + \lambda \|x\|_1 + \langle \xi, Ax - \tau \rangle \\ &= \max_{\xi \in \mathbb{R}^m} \Big[\min_{\tau \in \mathbb{R}^m} \{\|\tau - b\|^2 - \langle \xi, \tau \rangle\} + \min_{x \in \mathbb{R}^n} \{\lambda \|x\|_1 + \langle A^T \xi, x \rangle\} \Big] \\ &:= \max_{\xi \in \mathbb{R}^m} T_1(\xi) + T_2(\xi) \end{aligned}$$

As we made the substitution $\tau = Ax$, we are now able to separate the problems into $T_1(\xi)$ and $T_2(\xi)$. We work on $T_1(\xi)$ and referring to 2.6, we have

$$T_1(\xi) = -\frac{1}{2}\|\xi\|^2 + \langle \xi, b \rangle$$

We consider $T_2(\xi)$ coordinate-wise. For every $i \in \{1, \ldots, n\}$, we solve

$$\begin{aligned} T_{2,i}(\xi) &:= \min_{x_i \in \mathbb{R}} \lambda |x_i| + (A^T \xi)_i x_i \\ &= \min_{x_i \in \mathbb{R}} (\lambda + \operatorname{sgn}(x_i)(A^T \xi)_i) |x_i| \end{aligned}$$

Then we consider separate cases where,

$$T_{2,i}(\xi) = \begin{cases} 0, & \text{if } |(A^T \xi)_i| \leq \lambda, \\ \infty, & \text{otherwise} \end{cases}$$

Hence,

$$T_2(\xi) = \sum_{i=1}^{n} T_{2,i}(\xi) = \begin{cases} 0, & \text{if } \|A^T\xi\|_\infty \leq \lambda \\ \infty, & \text{otherwise} \end{cases}$$

Putting the pieces together we have,

$$D = \max_{\xi \in \mathbb{R}^m} -\frac{1}{2}\|\xi\|^2 + \langle \xi, b \rangle : \|A^T\xi\|_\infty \leq \lambda$$

Appendix B

Code Example

The following code can be accessed in our Github repository below, https://github.com/johnnymdoubleu/lassoSSNAL

We first call or install the necessary libraries.

```
library(rmatio[1])   # read in .mat format files
library(Rcpp[2])     # source C++ files
library(RSpectra[3]) # computes the eigen values
library(Matrix)   # handles matrix operations
```

Now, we source the **R** and **C**++ files that comprise our algorithm

```
source("Classic_Lasso_SSNAL.R")
source("Classic_Lasso_SSNAL_main.R")
source("Classic_Lasso_SSNCG.R")
source("proj_inf.R")
source("linsyssolve.R")
source("findstep.R")
source("psqmry.R")
source("matvec_ClassicLasso.R")
source("findnnz.R")
sourceCpp("mex_matrix_mult.cpp")
sourceCpp("mexsigma_update_classic_Lasso_SSNAL.cpp")
```

The following code lines now operates the SSNAL algorithm.

```
# load the data and split them into response and
# explanatory variables for Lasso regression.
data <- read.mat("UCIdata/abalone_scale_expanded7.mat")
A <- data$A
b <- data$b
```

[1] https://github.com/stewid/rmatio
[2] https://github.com/RcppCore/Rcpp
[3] https://github.com/yixuan/RSpectra

```
# defining the necessary arguments for the function
eps <- 2.220446e-16 # Copy the MATLAB eps essentially
n <- ncol(A)
c <- 10^(-4) # 10^(-3)
rho <- c * max(abs(t(t(b) %*% A)))
# evaluate the Lipschitz condition
lipfun <- function(b, A){
  return(t(t(A%*%b) %*% A))
}
eigs_AtA <- eigs_sym(lipfun, k = 1, n = n, args = A)

# set running parameters for R-SSNAL
opts <- c()
opts$stoptol <- 1e-6
opts$Lip <- eigs_AtA$values
opts$Ascale <- 1
opts$maxiter <- 1000

# running the algorithm and outputs with total profiling
Rprof()
clo <- Classic_Lasso_SSNAL(A, b, n, rho, opts)
Rprof(NULL)
summaryRprof()

# portion of the output values
cat("min(X) = ", clo$info$minx, "\n")
cat("max(X) = ", clo$info$max, "\n")
cat("nnz = ", findnnz(clo$info$x,0.999)$k, "\n")
```