Variable selection and outlier detection viamixed integer programming

Mahdi Jammal

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de l'INSA de Rouen et l'Université Libanaise

Variable Selection and Outlier Detection via Mixed Integer Programming

Présentée et soutenue par

Mahdi Jammal

Thèse soutenue publiquement le 17 décembre 2020 devant le jury composé de		
Marianne Clausel	Professeure à L'Université de Lorraine, France	Rapporteure
Patrick Siarry	Professeur à l'Université Paris-Est Créteil, France	Rapporteur
Sourour Elloumi	Professeure à l'École Nationale Supérieure de Techniques Avancées, France	Examinatrice
Stéphane Canu	Professeur à l'INSA de Rouen Normandie, France	Directeur de Thèse
Hassan Ibrahim	Professeur à l'Université Libanaise, Liban	Co-directeur de Thèse
Maher Abdallah	Professeur associé à l'Université Libanaise, Liban	Co-encadrant

Thèse dirigée par Stephane Canu et co-dirigée par Hassan Ibrahim

INSA ROUEN NORMANDIE

MIIS

litis

الجامعة اللبنانية
UNIVERSITE LIBANAISE

Table of contents

Abstract

Two principles at the forefront of modern machine learning and statistics are sparse modeling and robustness. Sparse modeling enables the construction of simpler statistical models. At the same time, statistical models need to be robust they should perform well when data is noisy in order to make reliable decisions.

While sparsity and robustness are often closely related, the exact relationship and subsequent trade-offs are not always transparent. For example, convex penalties like the Lasso are often motivated by sparsity considerations, yet the success of these methods is also driven by their robustness. In this thesis, we work at improving the quality of the estimators in supervised machine learning in terms of robustness and sparsity. We propose methods that, jointly, perform variable selection and outlier detection, and formulate the obtained optimization problems using mixed integer programming (MIP) benefiting from the significant improvement of MIP solvers.

First we focus on proposing a robust and sparse method for linear regression. To solve the problem exactly, we recast it as a mixed integer programming problem. In addition, and in order to decrease the computational time, we propose a discrete first algorithm providing a near-optimal solution in a very short time. The obtained solution is used as a warm start for the MIP solver. However, the proposed problem suffered from overfitting for low signal-to-noise ratio values.

Then, to fix the overfitting behavior of the proposed method, we use penalized regularization to improve its performance when the noise is high. We also propose a discrete first order algorithm to solve the regularized approach.

Finally, we propose a robust and sparse classification method based on the classical hinge-loss classifier. The obtained problem is formulated using mixed integer programming and shown to be efficient on both real and synthetic data sets.

Keywords : Variable selection, outlier detection, mixed integer programming, robust regression, sparse regression, robust support vector machines, sparse support vector machines.

Résumé

Deux points clés de l'apprentissage machine et des statistiques modernes sont la modélisation parcimonieuse et la robustesse. La modélisation parcimonieuse permet la construction de modèles statistiques plus performant et économes en ressources. Dans un même temps, les modèles statistiques doivent être robustes ; ils doivent être performants lorsque les données sont bruitées afin de permettre de prendre des décisions fiables.

Si la parcimonie et la robustesse sont souvent étroitement liées, la relation qui les unit et les compromis qui en découlent ne sont pas toujours explicitement formulés. Par exemple, des pénalités convexes comme celle du Lasso sont souvent motivés par des considérations de sélection de variable, mais le succès de ces méthodes est aussi dû à leur robustesse. Dans cette thèse, nous travaillons à l'amélioration de la qualité des estimateurs dans le cadre de l'apprentissage supervisé simultanément en termes de robustesse et de parcimonie. Nous proposons des méthodes qui réalisent simultanément les deux tâches de sélection de variables et de détection de points aberrants. Ce problème est formulé dans différents contextes sous la forme de problèmes d'optimisation utilisant la programmation mixte en nombres entiers (MIP), dont la résolution bénéficie de l'amélioration significative des solveurs MIP.

Nous nous concentrons d'abord sur la proposition d'une méthode robuste et peu répandue pour la régression linéaire. Pour résoudre le problème exactement, nous le reformulons en un problème de programmation mixte en nombres entiers. Ensuite, afin de réduire le temps de calcul, nous proposons un premier algorithme discret fournissant une solution quasi optimale en un temps très court. La solution obtenue est utilisée comme un démarrage à chaud pour le solveur MIP. Cependant, la solution proposée souffre de surapprentissage pour de faibles valeurs du rapport signal/bruit. Afin de corriger ce surapprentissage, nous utilisons une régularisation pénalisée pour améliorer ses performances lorsque le bruit est élevé. Nous proposons également un algorithme discret du premier ordre pour résoudre l'approche régularisée.

Enfin, nous proposons une méthode de classification robuste parcimonieuse basée sur le classificateur classique séparateur à vaste marge (SVM) associé à la fonction de perte «charnière ». Le problème obtenu est là aussi formulé comme un programme mixte en nombres entiers et s'avère efficace sur des ensembles de données réelles et synthétiques.

Mots-clés : Sélection de variables, points aberrants, programmation mixte en nombres entiers, régression robuste, régression parcimonieuse, séparatuer à vaste marge robustes, séparatuer à vaste marge parcimonieux.

Acknowledgment

First and foremost, I would like to thank my supervisors Dr. Stepahne Canu, Dr. Hassan Ibrahim and Dr. Maher Abdallah for their immense help and support not only through my graduate study but also for their guidance. Without them, this thesis would not be possible. I feel privileged to have had the opportunity to work and learn from them.

I would also like to thank Dr. Ali Wehbe and Dr.Ahmad Faour for all their support and help.

I would like to especially thank the "Islamic Center Association For Guidance and Higher Education" in Beirut for their grant program.

I am grateful to Brigitte, Sandra and Zeinab for all their administrative help.

A special thanks goes to the members of my dissertation committee : Maher Abdallah, Stephane Canu, Marianne Clausel, Sourour Elloumi, Hassan Ibrahim and Patrick Siarry for generously offering their time, support, guidance and good will throughout the preparation and review of this document.

Finally, my thanks go to my parents because thanks to them my professional project continues to be realized, to my friends for their support and encouragement throughout my present.

Chapitre 1

Introduction

1.1 General Framework

Artificial Intelligence (AI) has been a subject of great interest for many decades, aiming at making machines reproduce more or less specific human behaviors, ranging from playing chess to producing medical diagnostics. Specifically, in the past decade, this domain has seen a rapid and almost exponential growth, being invested by numerous research labs and companies (like Google, Facebook, Microsoft, Amazon, etc.). Nowadays, applications of AI are varied and show very impressive results. Those include information and image retrieval (web and image search engines), automatic translation, image recognition and classification (e.g. Google Photos), face recognition, tracking of objects in videos, speech recognition, making autonomous vehicles drive, interpreting medical imagery, etc.

Machine Learning (ML) is a sub-field of artificial intelligence, that addresses the following question: 'How can we program systems to automatically learn and to improve with experience?' Learning in this context is not learning by heart but recognizing complex patterns and making intelligent decisions based on data. Machine Learning uses algorithms to assist computer systems in progressively improving their performance. These algorithms automatically build a mathematical model using sample data to make decisions without being specifically programmed to make those decisions. Initially, ML was based on a model of brain cell interaction. The model was created in 1949 by [Heb49]. Recently, Machine Learning was defined by Stanford University as "the science of getting computers to act without being explicitly programmed." Machine Learning is now responsible for some of the most significant advancements in technology, such as the new industry of self-driving vehicles. Machine Learning can be mainly divided into two subcategories: supervised and unsupervised learning. Supervised learning is simply a process of learning algorithm from the training data set.

It is where you have input variables and an output variable and you use an algorithm to learn the mapping function from the input to the output. The aim is to approximate the mapping function so that when we have new input data we can predict the output variables for that data. As we can see in Figure 1.1, supervised learning consists of regression and classification. Regression algorithms attempt to estimate the mapping function f from the input variables X to numerical or continuous output variables y. In this case, y is a real value, which can be an integer or a floating point value. Therefore, regression prediction problems are usually quantities or sizes. For example, when provided with a data set about houses, and you are asked to predict their prices, that is a regression task because price will be a continuous output. On the other hand, classification algorithms

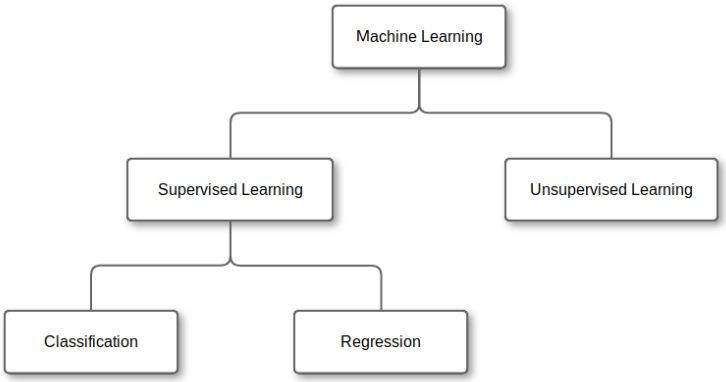

FIGURE 1.1 – Types of Learning.

attempt to estimate the mapping function f from the input variables X to discrete or categorical output variables y. In this case, y is a category that the mapping function predicts. If provided with a single or several input variables, a classification model will attempt to predict the value of a single or several conclusions.

For example, when provided with a data set about houses, a classification algorithm can try to predict whether the prices for the houses "sell more or less than the recommended retail price." Here, the houses will be classified whether their prices fall into two discrete categories: above or below the said price.

Unsupervised learning is modeling the underlying or hidden structure or distribution in the data in order to learn more about the data. In Unsupervised learning you only have input data and no corresponding output variables, it is done in the context of clustering. Figure 1.2 sheds the light on the difference between supervised and unsupervised learning.

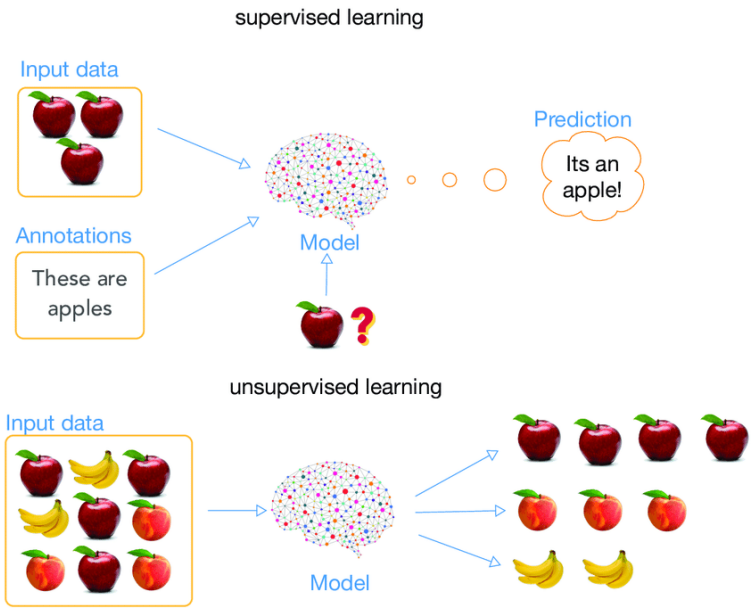

FIGURE 1.2 – Supervised vs Unsupervised Learning [MLG+18].

Today, people live in a world surrounded by diverse data. Improving technologies resulted an exponential growth of the data dimensionality and large volume data of high

dimension is produced every day. Large data sets will cause a difficult interpretation of the machine learning model. In addition, it can also cause bad performance of the model. However, it is noticed that in many machine learning applications, the number of information that makes sense is very small comparing to the whole data, that is to say, the underlying model is sparse. Also, outliers can have a considerable bad influence on estimators where outliers are understood to be observations that have been corrupted, incorrectly measured, mis-recorded, drawn under different conditions than those intended, or so atypical as to require separate modeling, see for instance [Wei05].

The natural way to deal with spurious features is to enumerate all possible combinations and choose the most significant features. Similarly, the detection of outliers can be made by enumerating all possible point configurations and neglect useless observations. Hence the common point is the counting function which is defined by:

$$\begin{array}{rccl} c: & \mathbb{R}^p & \longrightarrow & \mathbb{R} \\ & w & \longmapsto & c(w) = \text{ the number of nonzero components } w_i \text{ of } w. \end{array}$$

It is often called, by an abuse of language, the ℓ_0 norm and denoted by $c(w) = ||w||_0$. In linear regression, feature selection using the ℓ_0 dates back to at least [BKM67, HL67], this problem is called "best subset selection problem". Since the ℓ_0 norm is not convex neither differentiable, this problem was considered NP-hard and intractable. Hence to overcome its computational difficulty, the ℓ_0 norm was replaced by convex surrogates such as the ℓ_2 norm known as ridge regression [HK70] or by the ℓ_1 norm known as Lasso regression [Tib96].

Outlier detection can be divided into two categories: (a) the so-called "robust statistics", that is robust-to-outlier loss functions and (b) outlier detection per se which excludes outliers from the training set. One can add a variable modeling outliers and can use the ℓ_0 norm to control its sparsity level. Hence, selecting variables and detecting outliers can be done by adding two cardinality constraints representing the number of variables to be selected and the number of outliers to be detected.

Similarly, support vector machine (SVM) classification suffers from irrelevant features and outliers. To this end, many approaches were proposed. However, to the best of our knowledge, this is the first time that the ℓ_0 norm is used for both operations.

1.2 Motivations

In the last three decades (1990-2020), algorithmic advances in integer optimization combined with hardware improvements have resulted in an astonishing 800 billion factor speedup in solving Mixed Integer Optimization (MIO) problems. Indeed, between 1991 and 2015 the software total speedup is about 1.4 million times and the hardware total speedup is about 570000 times. Hence a mixed integer optimization problem that would have taken 26000 years to solve 25 years ago can now be solved in a modern computer in less than one second: *mixed integer linear techniques are nowadays mature, that is fast, robust, and are able to solve problems with up to millions of variables* [LL11].

As it is known, originally, the sparsity is explicitly expressed by using the ℓ_0 norm. This function can exactly control the sparsity level, however, with a shortcoming of non-convexity and non-differentiability, which makes it the obstacle to overcome. The existing methods for solving this problem can be roughly grouped in two major classes: Greedy algorithms attains the solution by iteratively providing suboptimal solutions, such as matching pursuit (MP) and its variants [iG04, BRF11], or gradient descent based algorithms

such as Iterative Hard Thresholding (IHT) algorithm [GK09] or proximal method [BJQS14]. The second class corresponds to methods that relax the ℓ_0 norm by replacing it with the ℓ_1 norm, which is still nonsmooth but convex and continuous. This leads to a classical problem, often called the problem of LASSO. However, the ℓ_1 norm cannot always guarantee the required sparsity level and even the sparse representation is produced but with shrinkage.

When referring to the problem of feature selection and outlier detection for linear regression, it corresponds to adding two ℓ_0 norm constraints, which makes it even more complex than the best subset selection problem consisting only on one cardinality constraint. To reach tho optimal solution, we reformulate the original problem as mixed integer quadratic programming (MIQP) problem and solve it using an efficient solver "Gurobi". To accelerate the MIQP solution, we introduce a warm start solution obtained by an algorithm based on a discrete extension of first order continuous optimization methods.

This framework is scalable and provides a local solution often close to the global one. Results on both real and synthetic datasets showed the high quality of the provided solutions. Furthermore, since the ℓ_0 norm suffers from overfitting behavior when the signal to noise ratio is low (high noise level in data), we also propose an ℓ_1 regularized robust and sparse regression model. We solve this problem exactly using mixed integer programming, and we propose a first order discrete algorithm which can be used as a warm start for the MIO solver. The first order algorithm can also be used for high dimensional data sets since it provides solutions with good predictive performance as it is shown by empirical results.

Regarding the support vector machine (SVM), to deal with outliers, a new variable is added to the classical hinge-loss support vector machine formulation. Furthermore, to ensure sparsity and to control the percentage of outliers to be detected, two cardinality constraints were added. The obtained problem is reformulated as a mixed integer optimization problem to obtain the global solution. Unlike all other methods, the MIO approach always provides a guarantee on its sub-optimality even if the algorithm is terminated early. Empirical results on both synthetic and real datasets revealed the significant improvement in comparison with the 0-1 loss and hinge-loss classical classification problems.

1.3 Outline and Contributions

This monograph concentrates on the study of two issues encountered in supervised learning: the presence of irrelevant features and the presence of outliers. More precisely, dimension reduction or feature selection is an effective strategy to handle contaminated data and to deal with high dimensionality while providing better prediction. Also, outliers, which are understood to be observations that have been corrupted or incorrectly measured or drawn under different conditions than those intended, can have a considerable bad influence on estimators.

To deal with outlier proneness and spurious variables, a new variable representing the outliers was added to the objective function, and two ℓ_0 norm constraints were added to control the number of variables to be selected and the number of outliers to be detected. The obtained optimization problems are $\mathcal{NP}$-hard. To obtain the global solution exactly, we recasted these problems as mixed integer programming problems benefitting from the significant improvement in the speed of solving such problems by using efficient solvers such as "Gurobi".

This impressive speedup factor is due to incorporating both theoretical and practical

advances into mixed integer programming (MIP) solvers. Cutting plane theory, disjunctive programming for branching rules, improved heuristic methods, techniques for preprocessing MIPs, using linear optimization as a black box to be called by MIP solvers, and improved linear optimization methods have all contributed greatly to the speed improvements in MIP solvers. MIP solvers in general, and Gurobi in particular, provide both feasible solutions as well as lower bounds to the optimal value. As the MIP solver progresses towards the optimal solution, the lower bounds improve and provide an increasingly better guarantee of suboptimality, which is especially useful if the MIP solver is stopped before reaching the global optimum. Incontrast, heuristic methods do not provide such a certificate of suboptimality. Furthermore, we developed a discrete extension of modern first order continuous optimization methods to find high quality feasible solutions that we use as warm starts to the MIP solver that finds provably optimal solutions.

The rest of the monograph is organized as follows.

Chapter 2 introduces a mathematical background on optimization. All needed definitions and optimizations methods were briefly described and presented.

Chapter 3 presents our robust and sparse linear regression problem. We use MIP to find a provably optimal solution for the obtained problem. Our approach has the appealing characteristic that if we terminate the algorithm early, we obtain a solution with a guarantee on its suboptimality. Furthermore, our framework can accommodate side constraints on the variables. We introduce a general algorithmic framework based on a discrete extension of modern first order continuous optimization methods that provide near-optimal solutions for the best subset problem. The MIP algorithm significantly benefits from solutions obtained by the first order methods. Computational results on both real and synthetic data sets were reported. These results show that the proposed framework provides high quality solutions.

Chapter 4 aims to deal with a problem encountered in Chapter 3: the robust and sparse linear regression model underperforms for low signal-to-noise ratio values because of the overfitting behavior of the ℓ_0 norm. To reduce this overfitting behavior, a regularization ℓ_1 term was added to the objective function. The obtained optimization problem was recasted as a mixed integer program and a discrete first order algorithm was proposed to find near optimal solutions for a low computational cost. Results on synthetic data sets show that the regularized problems overperformed the ℓ_0 methods for low SNR values and as the SNR increases, these methods become almost similar. Furthermore, results on real data sets show that the proposed method produces high quality solutions.

In Chapter 5 the problem of both feature selection and outlier detection using the ℓ_0 norm in classification is addressed. The proposed approach performs, jointly, the selection of relevant variables and the detection of outliers. The problem is formulated as a mixed integer program which allows the use of the efficient solver "Gurobi" to solve it. The proposed approach is compared with the classical 0-1 loss and hinge loss classification problems. The results show that the proposed approach provides high quality solutions.

In Chapter 6 we conclude this thesis and discuss several interesting directions for future work.

1.4 Publications

The work contained in this thesis will be published or submitted for publication in the following papers:

— Mahdi Jammal, Stephane Canu and Maher Abdallah. "Robust and Sparse Support Vector Machines via Mixed Integer Programming". In Proceedings of *The Sixth International Conference on Machine Learning, Optimization, and Data Science – July 19-23, 2020 – Certosa di Pontignano, Siena – Tuscany, Italy*. To be published in *Lecture Notes in Computer Science Volume 12514 - Springer.*

— Mahdi Jammal, Stephane Canu and Maher Abdallah. "ℓ_1 Regularized Robust and Sparse Linear Modeling Using Discrete Optimization". In Proceedings of *The Sixth International Conference on Machine Learning, Optimization, and Data Science – July 19-23, 2020 – Certosa di Pontignano, Siena – Tuscany, Italy*. To be published in *Lecture Notes in Computer Science Volume 12514 - Springer.*

— Mahdi Jammal, Stephane Canu and Maher Abdallah. "Joint Outlier Detection and Variable Selection Using Discrete Optimization". *Submitted to the Journal of Global Optimization.*

Chapitre 2

Optimization Tools for Machine Learning

Sommaire

2.1 Introduction

Since its earliest days as a discipline, machine learning has made use of optimization formulations and algorithms. Likewise, machine learning has contributed to optimization, driving the development of new optimization approaches that address the significant challenges presented by machine learning applications. This cross-fertilization continues to deepen, producing a growing literature at the intersection of the two fields while attracting leading researchers to the effort.

Optimization approaches have enjoyed prominence in machine learning because of their wide applicability and attractive theoretical properties. While techniques proposed twenty years and more ago continue to be refined, the increased complexity, size, and variety of today's machine learning models demand a principled reassessment of existing assumptions and techniques.

This Chapter makes a start toward such a reassessment. Besides describing the resurgence in novel contexts of established frameworks such as first order methods, proximal

methods the section devotes significant attention linear programming and mixed integer programming needed in the rest of the thesis.

Of course many other useful optimization tools have been used in machine learning such as heuristic and metaheuristic algorithms for combinatorial optimization. Because this work initially focuses on the exact solutions to optimization problems, these approaches were not considered in this thesis.

For further mathematical details, we recommend the following references [SNW12, NW06, BV04, WN99] from which the writing of this chapter is largely inspired.

2.2 Mathematical background

In machine learning in general and in this thesis in particular, we are usually dealing with real variables of high dimension n. To this end, we will focus on optimization problems over the set $\mathbb{R}^n$ in the rest of the Chapter.

Consider a mathematical optimization problem of the form:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \quad i = 1,\dots,m, \end{cases} \tag{2.1}$$

where $x = (x_1,\dots,x_n)^\top$ is the vector of optimization variable of the problem. The function $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the given objective function and functions $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ are the given inequality constraint functions. A vector $x^\star$ is called an optimal solution or global minimizer to (2.1) if it has the smallest objective value among all vectors satisfying the constraints, that is, for any z with $f_i(z) \leq 0$ $i = 1,\dots,m$ we have $f_0(z) \geq f_0(x^\star)$.

Suppose $x_1 \neq x_2$ are two points in $\mathbb{R}^n$. Points of the form $y = \theta x_1 + (1-\theta)x_2$ where $\theta \in \mathbb{R}$, form the line passing through x_1 and x_2. The parameter value $\theta = 0$ corresponds to $y = x_2$, and the value $\theta = 1$ corresponds to $y = x_1$. Values of parameter θ between 0 and 1 corresponds to the closed line segment between x_1 and x_2.

Definition 2.2.1 (Convex set) *A set* C *is convex , if the line segment between any two points in* C *lies in* C *i.e if for any* $x_1, x_2 \in$ C *and any* $\theta \in \mathbb{R}$ *with* $0 \leq \theta \leq 1$*, we have:*

$$\theta x_1 + (1-\theta)x_2 \in \text{C}.$$

The notion of convex set is illustrated Figure 2.1. The domain or set of departure of a function is the set into which all of the input of the function is constrained to fall. It is the set X in the notation $f : \text{X} \mapsto \text{Y}$, and is alternatively denoted as $dom(f)$.

Definition 2.2.2 (Convex function) *A function* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ *is convex if* $dom(f)$ *is a convex set and if for all* $x, y \in dom(f)$*, and* $\theta \in \mathbb{R}$ *with* $0 \leq \theta \leq 1$*, we have*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y).$$

This inequality means that the line segment between $(x, f(x))$ *and* $(y, f(y))$ *lies above the graph of* f*.*

This definition of convex function is illustrated Figure 2.2.

Definition 2.2.3 (Local minimizer) *A vector* $x^\star \in \mathbb{R}^n$ *is called a local minimizer of the optimization problem* $\min\limits_{x\in\text{C}} f(x)$*, if there exists a neighbourhood* B *of* $x^\star$ *such that* $x^\star$ *is a global minimizer of the problem* $\min\limits_{x\in\text{C}\cap\text{B}} f(x)$*.*
In another way, $x^\star$ *is a local minimizer if there exists* $\epsilon > 0$ *such that:*

$$f(x^\star) < f(x) \text{ for all } x \in \text{B}(x^\star,\epsilon) := \{x \in \mathbb{R}^n : ||x - x^\star|| \leq \epsilon\}.$$

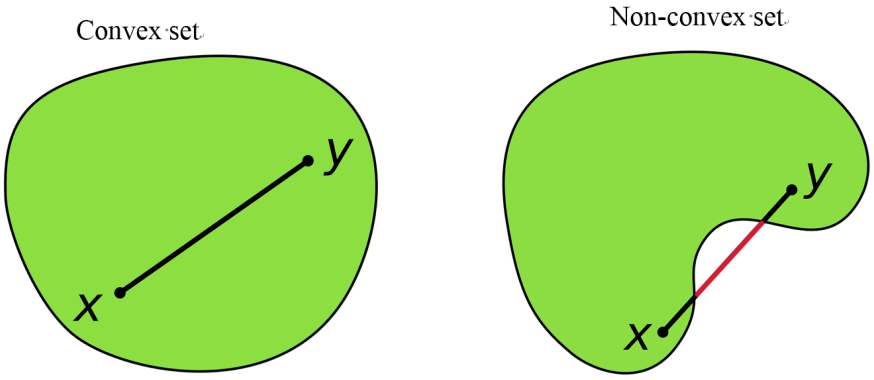

FIGURE 2.1 – The kidney shaped set (Right) is non-convex since the line segment between the two points x and y in the set (taken from [BV04]).

In what follows, $x, y \in \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}$.

Theorem 2.2.1 (First order optimality condition) *Suppose f is differentiable (i.e its gradient ∇f exists at each point in $dom(f)$). Then f is convex if and only if $dom(f)$ is convex and*

$$f(y) \geq f(x) + \nabla f(x)^{\mathrm{T}}(y - x). \tag{2.2}$$

holds for all $x, y \in dom(f)$.

The inequality above shows that from a local information about a convex function (its value and its derivative at a point), we can derive a global information. This is perhaps the most important property of convex functions.

Theorem 2.2.2 *If $\hat{x}$ is a local minimizer of a convex optimization problem, then it is a global minimizer.*

Theorem 2.2.3 (Second order optimality condition) *Assume that f is twice differentiable, that is its hessian or second derivative $\nabla^2 f$ exists at each point in $dom(f)$. Then f is convex if and only if $dom(f)$ is convex and its hessian is positive semi-definite, that is, for all $x \in dom(f)$:*

$$\nabla^2 f(x) \succcurlyeq 0.$$

Definition 2.2.4 (Subgradient) *g is a subgradient of a convex function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ at $x \in dom(f)$, if*

$$f(z) \geq f(x) + g^{\mathrm{T}}(z - x) \quad \forall z \in dom(f).$$

FIGURE 2.2 – Example of a Convex Function.

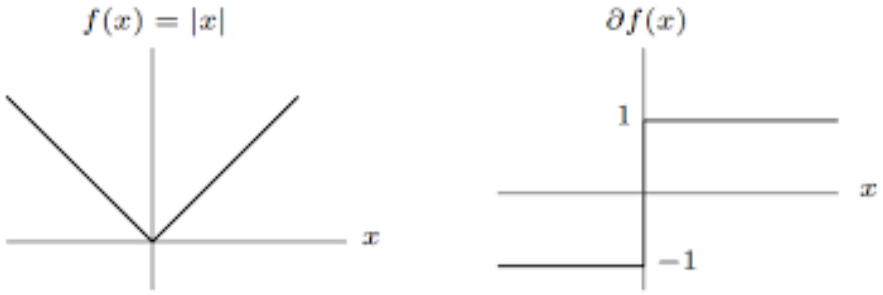

FIGURE 2.3 – The absolute value function (left), and its subdifferential $\partial f(x)$ as a function of x (right) [BDV11].

This definition of subgradient is illustrated Figure 2.4.

If f is convex and differentiable, then its gradient at x is a subgradient. However, a subgradient can exist even when f is not differentiable at x, as illustrated in figure 2.4. For more details about subgradient, see [BDV11].

Definition 2.2.5 (Subdifferential) *The set of all subgradients of f at x is called the subdifferential. It is denoted by $\partial f(x)$.*

We note that if f is convex and differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$, i.e., its gradient is its only subgradient. Conversely, if f is convex and $\partial f(x) = \{g\}$, then f is differentiable at x and $g = \nabla f(x)$.

Example: consider $f(x) = |x|$. For $x < 0$, the subgradient is unique: $\partial f(x) = \{-1\}$. Similarly, for $x > 0$, the subgradient is unique: $\partial f(x) = \{+1\}$. At $x = 0$, the subdifferential is defined by the inequality $|z| \geq gz$ for all $z \in \mathbb{R}$, which is satisfied if and only if $g \in [-1, 1]$. Therefore, we have $\partial f(0) = [-1, 1]$. This example is illustrated Figure 2.3.

Theorem 2.2.4 (Minimize of a non-differentiable convex function) *A point $x^\star$ is a minimizer of a convex function f if and only if f is subdifferential at $x^\star$ and*

$$0 \in \partial f(x^\star),$$

i.e $g = 0$ is a subgradient of f at $x^\star$. This condition reduces to $\nabla f(x^\star) = 0$ if f is differentiable at $x^\star$.

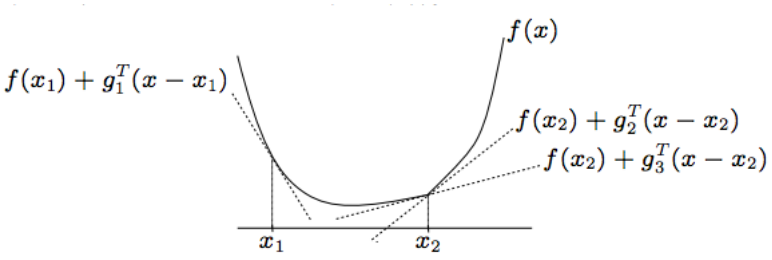

FIGURE 2.4 – At x_1, the convex function f is differentiable, and g_1 (which is the derivative of f at x_1) is the unique subgradient at x_1. At the point x_2, f is not differentiable. At this point, f has many subgradients: two subgradients, g_2 and g_3, are shown [BDV11].

For more details and proofs of the previous theorems, see for instance [SNW12, BV04].

2.3 Convex Optimization problems

A convex optimization problem is a problem consisting of minimizing a convex function over a convex set. More explicitly, a convex problem is of the form:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{subject to} & x \in \mathrm{C}, \end{cases} \tag{2.3}$$

where C is a convex set and f is a convex function over C.

2.3.1 Unconstrained convex and differentiable optimization problems

Consider the following convex optimization problem:

$$\min_x f(x), \tag{2.4}$$

where $f(x)$ is a convex function. This problem is an unconstrained optimization problem. We recall that if $x^\star$ is a local minimizer of this problem, then $x^\star$ is also a global minimizer. This means that a necessary and sufficient condition for $x^\star$ to be the solution of problem (2.4) is that, when f is differentiable and ∇f exists , $\nabla f(x^\star) = 0$, where $\nabla f(x)$ is the gradient vector of $f(x)$, that is

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}. \tag{2.5}$$

Algorithms to solve unconsrtained optimization problems

We will shortly present two examples of efficient algorithms used to solve unconstrained convex optimization problems.

Gradient descent algorithm

We know that in order to minimize a convex function, we need to find a stationary point. There are different methods and heuristics to find a stationary point. One possible approach is to start at an arbitrary point, and move along the gradient at that point towards the next point, and repeat until (hopefully) converging to a stationary point.

In general, one can consider a search for a stationary point as having two components: the direction and the step size. The direction decides which direction we search next, and the step size determines how far we go in that particular direction. Such methods can be generally described as starting at some arbitrary point $x^{(0)}$ and then at every step $k \leq 0$ iteratively moving at direction $\Delta x^{(k)}$ by step size t_k to the next point $x^{(k+1)} = x^{(k)} + t_k \Delta x^{(k)}$. In gradient descent, the direction we search is the negative gradient at the point, i.e $\Delta x = -\nabla f(x)$. Thus, the iterative search of gradient descent can be described through the following recursive rule:

$$x^{(k+1)} = x^{(k)} - t_k \nabla f(x^{(k)}).$$

Since our objective is to minimize the function, one reasonable approach is to choose the step size in manner that will minimize the value of the new point i.e find the step size

that minimizes $f(x^{(k+1)})$. The step size $t_k^\star$ of this approach is optimal and given by:

$$t_k^\star = \underset{t\geq 0}{\operatorname{argmin}} f(x^{(k)} - t\nabla f(x^{(k)})).$$

Algorithm 2.1 presents the pseudo code of the gradient descent algorithm with optimal stepsize.

Algorithme 2.1 : Gradient descent.

Input : Guess $x^{(0)}$, set $k = 0$
Output : x^{k+1}
while $||\nabla f(x^{(k)})|| \geq \epsilon$ **do**
    $t_k = \underset{t\geq 0}{\operatorname{argmin}} f(x^{(k)} - t\nabla f(x^{(k)}))$.
    $x^{(k+1)} = x^{(k)} - t_k \nabla f(x^{(k)})$
    $k \leftarrow k+1$

The advantages of the gradient descent algorithm can be summarized as follows:

— each iteration is inexpensive ;

— does not require second derivative ;

while the drawbacks can be summarized as follows:

— often slow and sensitive to scaling ;

— does not handle non differentiable functions.

Newton's method

The gradient descent method uses only first derivatives in selecting a suitable search direction. However, Newton's method uses first and second derivatives and indeed preforms better.

To solve an unconstrained optimization convex problem, we substitute the direction $-\nabla f(x)$ by the direction $-\mathrm{H}(x)^{-1}\nabla f(x)$ where $\mathrm{H}(x) = \nabla^2 f(x)$ is the hessian of f at x, that is the square matrix of its second order partial derivatives.

2.3.2 Methods for non-differentiable functions

When the objective function f is non-differentiable many methods exists to solve the corresponding optimization problem. The most popular ones in machine learning are the subgradient method and the proximal gradient descent that we will now briefly introduce.

Subgradient method

To minimize a non-differentiable convex function f, substitute $\nabla f(x)$ of the gradient descent method, by $g(x)$, where $g(x)$ is any subgradient of f at x. The subgradient method has a disadvantage in that it can be much slower than interior-point methods such as Newton's method, it as the advantage of the memory requirement being often times much smaller than those of an interior-point or Newton method, which means it can be used for extremely large problems for which interior-point or Newton methods cannot be used. Instead of trying to improve across the board, we will focus on minimizing composite functions of the form :

$$f(x) = g(x) + h(x),$$

where g is convex, differentiable and $dom(g) = \mathbb{R}^n$. Also h is convex but not necessarily differentiable.

Proximal gradient descent

The proximal gradient descent algorithm is based on the use of the proximal operator defined as follows.

Definition 2.3.1 (The proximal operator) *The proximal operator of a convex function* $h : \mathbb{R}^n \to \mathrm{R}$ *at a point* $x \in \mathbb{R}^n$ *is defined by:*

$$prox_h(u) = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} (h(u) + \frac{1}{2}||u - x||_2^2).$$

Examples of proximal operators: [SNW12]

— $h(x) = 0$: $prox_h(x) = x$;

— $h(x) = \mathrm{I_C}(x)$ (indicator function of C): $prox_h$ is the projection on C, that is, $prox_h(x) = \mathrm{P_C}(x)$;

— $h(x) = ||x||_1$: $prox_h$ is the following soft thresholding operator

$$prox_h(x)_i = \begin{cases} x_i - 1 & if\ x_i \geq 1 \\ 0 & if\ |x_i| \leq 1 \\ x_i + 1 & if\ x_i \leq -1. \end{cases}$$

Consider a composite function f that can be decomposed as the sum of two functions g and h as follows:

$$f(x) = g(x) + h(x).$$

For a differentiable function f, the gradient update $x^+ = x - t\,\nabla f(x)$ is derived using quadratic approximation (replace $\nabla^2 f(x)$ by $\frac{1}{2}\mathrm{I}$):

$$x^+ = \underset{z}{\operatorname{argmin}}\ f(x) + \nabla f(x)^{\mathrm{T}}(z - x) + \frac{1}{2t}||z - x||_2^2.$$

For a decomposable function $f(x) = g(x) + h(x)$, if we use the quadratic approximation for the function g, then:

$$\begin{aligned} x^+ &= \operatorname{argmin}_z\ g(x) + \nabla g(x)^{\mathrm{T}}(z - x) + \tfrac{1}{2t}||z - x||_2^2 + h(z) \\ &= \operatorname{argmin}_z \tfrac{1}{2t}||z - (x - t\nabla g(x))||_2^2 + h(z) \\ &= prox_{h,t}(x - t\nabla g(x)). \end{aligned}$$

Therefore, the proximal gradient descent can be defined as follows:

— choose an initial guess $x^{(0)}$

— repeat $x^{(k)} = prox_{t_k}(x^{(k-1)} - t_k \nabla g(x^{(k-1)}),\ k = 1, 2, 3, \ldots$

Notes

— the proximal operator can be computed analytically for a lot of important h functions,

— the proximal operator does not depend on g at all, only on h,

— g can be a complicated function, all we need to do is to compute its gradient,

— the proximal algorithm is very sensitive to the choice of the stepsize sequence t_k.

2.4 Constrained Convex Optimization Problems

In the the optimization problem 2.3, if the constraint set C is a convex set and the objective function f is a convex function on C, then the problem is said to be a convex optimization problem. In this case, any local minimizer is also a global minimizer.

When C is described as the intersection of a finite number of inequality and equality constraints defined by some functions $g_i : \mathbb{R}^n \longrightarrow \mathbb{R},\ i = 1,\ldots,m$ and $h_j : \mathbb{R}^n \longrightarrow \mathbb{R},\ j = 1,\ldots,d$ as follows:

$$\mathrm{C} = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0,\ i = 1,\ldots,m\,; h_j(x) = 0,\ j = 1,\ldots,d\}.$$

The resulting constrained optimization problem is then written as:

$$\begin{cases} \min\limits_{x} & f(x) \\ s.t & g_i(x) \leq 0 \quad i = 1,\ldots,m \\ & h_j(x) = 0 \quad j = 1,\ldots,d. \end{cases}$$

If the objective function f and inequality constraint functions g_i are all convex and the equality constraint functions h_j are all affine, then the above problem is a convex optimization problem.

For convex constrained optimization problems, many specialized solution methods have been developed; often, these are tailored to different classes of optimization problems (such as linear programs, quadratic programs, semi-definite programs, etc).

Projected gradient descent

Projected gradient descent (PGD) is one type of general-purpose solution method for (convex) constrained optimization problems. The basic idea is quite simple. Consider a convex optimization problem with convex function f and convex set C. A standard gradient descent approach, which can be used to find a local minimum for an unconstrained optimization problem, is problematic here since the iterates x^k may fall outside the constraint set C. What PGD does is simply correct for this situation: on each iteration k, it first applies a standard gradient descent step to obtain an intermediate point $\tilde{x}^{k+1}$ that might fall outside C, and then projects this point back to the constraint set C by finding a point x^{k+1} in C that is closest (in terms of Euclidean distance) to$\tilde{x}^{k+1}$.

For convex C, the point x^{k+1} after projection is always closer to the minimizer in C than is the result of the gradient descent step $\tilde{x}^{k+1}$, and for suitable choices of the step-sizes, PGD converges to a (global) minimizer of the (convex) constrained optimization problem. Note that in order to implement PGD in practice, one must be able to efficiently compute projections onto the constraint set C; this can be done easily for certain types of constraint sets, but can be harder for others.

Algorithme 2.2 : Projected Gradient Descent.

Input : Guess $x^{(0)}$

for $k = 0,\ldots,\mathrm{T}$ **do**

$$\tilde{x}^{(k+1)} = x^{(k)} - t_k \nabla f(x^{(k)})$$
$$x^{k+1} = \underset{x \in \mathrm{C}}{\operatorname{argmin}} ||x - \tilde{x}^{(k+1)}||_2^2$$

Output : x^{k+1}

Lagrangian duality

Any constrained optimization problem has an associated (Lagrange) dual optimization problem, which is always convex. Sometimes, the dual problem is easier to solve, and can be used to obtain useful information about the original (primal) problem; in some cases, it can even be used to obtain a solution to the primal problem.
Consider the constrained optimization problem **??**. We will refer to it as the primal problem and we will denote its constraint set by C:

$$\mathrm{C} = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0,\ i = 1,\dots,m\,; h_j(x) = 0,\ j = 1,\dots,d\},$$

Any point $x \in \mathrm{C}$ is said to be primal feasible. Denote by $p^\star$ the optimal value of the primal problem:

$$p^\star = \inf_{x \in \mathrm{P}} f(x).$$

Note that $p^\star$ may or may not be achieved.
In Lagrangian duality, one introduces dual variables λ_i and ν_j associated with each of the inequality constraints $g_i(x) \leq 0$ and equality constraints $h_j(x) = 0$, respectively, and augments the objective function f by adding the constraint functions multiplied by the corresponding dual variables to obtain the Lagrangian function. Formally, the Lagrangian function $\mathscr{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^d \longrightarrow \mathbb{R}$ associated with the optimization problem **??** is defined as:

$$\mathscr{L}(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{d} \nu_j h_j(x).$$

The dual variable λ_i and ν_j are also referred to as the Lagrange multipliers associated with the inequality and equality constraints, respectively.

Definition 2.4.1 (Lagrange dual problem) *The Lagrange dual optimization problem is defined by the following optimization problem:*

$$\begin{cases} \max\limits_{\lambda,\nu} & \phi(\lambda, \nu) \\ s.t & \lambda_i \geq 0 \quad i = 1,\dots,m, \end{cases}$$

where

$$\phi(\lambda, \nu) = \inf_{x \in \mathbb{R}^n} \mathscr{L}(x, \lambda, \nu).$$

This dual problem is always a convex optimization problem, even if the primal problem is not convex (note that the dual function is a pointwise infimum of affine functions of $(\lambda; \nu)$, and therefore concave). Let $\mathrm{C_D}$ denote the constraint set of this dual problem:

$$\mathrm{C_D} = \{(\lambda, \nu) \in \mathbb{R}^m \times \mathbb{R}^d \mid \lambda_i \geq 0, i = 1,\dots,m\}.$$

Any point $(\lambda, \nu) \in \mathrm{C_D}$ is said to be dual feasible. Denote by $d^\star$ the optimal value of the dual problem:

$$d^\star = \sup_{(\lambda,\nu) \in \mathrm{C_D}} \phi(\lambda, \nu).$$

The optimal value of the dual problem always gives a lower bound on the optimal value of the primal problem. $d^\star \leq p^\star$.

Definition 2.4.2 (Duality gap) *The duality gap is the non-negative number $p^\star - d^\star$.*

Definition 2.4.3 (Strong duality) *We say that strong duality holds for problem if the duality gap is zero, that is:*

$$p^\star = d^\star.$$

Definition 2.4.4 (Slater's condition) *We say that the problem satisfies Slater's condition if it is strictly feasible, that is:*

$$\exists x_0 \in \mathrm{C_D} : g_i(x_0) < 0, i = 0, \ldots, m, \;\; h_j(x_0) = 0, j = 1, \ldots, d.$$

We can replace the above by a weak form of Slater's condition, where strict feasibility is not required whenever the function f_i is affine.
For further informations and details, we suggest the reference [WN88].

Theorem 2.4.1 (Strong duality via Slater condition) *If the primal problem is convex, and satisfies the weak Slater's condition, then strong duality holds, that is, $p^\star = d^\star$.*

Note that there are many other similar results that guarantee a zero duality gap.

For example:

Theorem 2.4.2 (Quadratic convex optimization problems) *If f_0 is quadratic convex, and the functions $f_1, \ldots, f_m, h_1, \ldots, h_d$ are all affine, then the duality gap is always zero, provided one of the primal or dual problems is feasible. In particular, strong duality holds for any feasible linear optimization problem.*

Example

Consider the following convex quadratic optimization problem over $\mathbb{R}^n$ with equality constraints only:

$$\begin{array}{ll} \min\limits_{x} & \frac{1}{2} x^{\mathrm{T}} x \\ s.t & \mathrm{C}x = d, \end{array}$$

where $\mathrm{C} \in \mathbb{R}^{d \times n}$ and $d \in \mathbb{R}^d$. Here Slater's condition (trivially) holds, and therefore we have strong duality. Introducing dual variables $\nu \in \mathbb{R}^d$, the Lagrangian function is given by:

$$\mathscr{L}(x, \nu) = \frac{1}{2} x^{\mathrm{T}} x + \nu^{\mathrm{T}} (\mathrm{C}x - d).$$

Minimizing over $x \in \mathbb{R}^n$ gives the followig dual problem:

$$\phi(\nu) = -\frac{1}{2} \nu^{\mathrm{T}} \mathrm{C}\mathrm{C}^{\mathrm{T}} \nu - d^{\mathrm{T}} \nu.$$

The dual optimization problem is then the following unconstrained convex quadratic program over $\nu \in \mathbb{R}^d$:

$$\max \; -\frac{1}{2} \nu^{\mathrm{T}} \mathrm{C}\mathrm{C}^{\mathrm{T}} \nu - d^{\mathrm{T}} \nu.$$

This dual problem does not depend on the number of variables n in the primal problem. Thus if the number of variables d in the primal problem is significantly larger than the number of equality constraints d, then solving the dual problem can be more efficient than solving the primal directly. On obtaining the dual solution $\nu^\star$, the primal solution is given by $x^\star = -\mathrm{C}^{\mathrm{T}} \nu^\star$.

Karush-Kuhn-Tucker (KKT) Optimality Conditions

If strong duality holds and the optimal values of the primal and dual problems are achieved at $x^\star$ and at $(\lambda^\star;\nu^\star)$, respectively, then the following 5 conditions, collectively called the Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} g_i(x^\star) &\le 0 \quad i=1,\dots,m && \text{(primal feasibility)}\\ h_j(x^\star) &= 0 \quad j=1,\dots,d && \text{(primal feasibility)}\\ \lambda_i^\star &\ge 0 \quad i=1,\dots,m && \text{(dual feasibility)}\\ \lambda_i^\star g_i(x^\star) &= 0 \quad i=1,\dots,m && \text{(complementary slackness)}\\ \nabla f(x^\star)+\sum_{i=1}^{m}\lambda_i^\star\nabla g_i(x^\star)+\sum_{j=1}^{d}\nu_j^\star\nabla h_j(x^\star) &= 0 && \text{(stationarity condition)} \end{aligned}$$

Under strong duality, the KKT conditions are necessary for optimality of $x^\star$ and $(\lambda^\star,\nu^\star)$. Furthermore, if the primal problem is convex, then the KKT conditions are also sufficient for points $x^\star$ and $(\lambda^\star,\nu^\star)$ to be optimal; see [BV04] for details. Thus in particular, if the primal problem is convex and Slater's condition holds, then the KKT conditions are both necessary and sufficient for optimality.

2.5 Non-convex optimization

Modern applications frequently require learning algorithms to operate in extremely high-dimensional spaces. Dealing with such high dimensionalities necessitates the imposition of structural constraints on the learning models being estimated from data. Such constraints are not only helpful in regularizing the learning problem, but often essential to prevent the problem from becoming ill-posed.

2.5.1 Non-Convex Projections and Hard-Thresholding Operator

Executing the projected gradient descent algorithm with non-convex problems requires projections onto non-convex sets.
A quick look at the projection problem:

$$\mathrm{P}_\mathrm{Q}(z)=\operatorname*{argmin}_{x\in\mathrm{Q}}||x-z||_2^2,$$

reveals that this is an optimization problem in itself. Thus, when the set Q is not convex, the projection problem itself can be NP-hard. However, for several well-structured sets, projections can be carried out efficiently despite the set being non-convex. We will only introduce the projections into sparse vectors since this projection will be used later.
Before entering into the details of the alternate projected gradient algorithm, it is appropriate to introduce the problem of finding the projection of a vector $u\in\mathbb{R}^n$ onto the set of $k\le n$ sparse vectors

$$\begin{aligned} \min_{v\in\mathbb{R}^n}\quad & \tfrac{1}{2}\|v-u\|^2\\ \text{s.t.}\quad & \|v\|_0\le k. \end{aligned} \tag{2.6}$$

This problem is easy and its solution $v^\star$ is given by sorting on the absolute value of vector $|u|$, that is by a sequence of indices (j) such that $|u_{(1)}|\ge|u_{(2)}|\ge\ldots|u_{(j)}|\ge\cdots\ge|u_{(n)}|$. Using these indices, the projection $v^\star=\mathrm{P}_k(u)$ of u is the vector u itself with its smallest coefficients set to 0 that is

$$v^\star=\mathrm{P}_k(u)=\begin{cases} u_j & \text{if } j\in\{(1),\dots,(k)\}\\ 0 & \text{else.}\end{cases}$$

This operator is known as the hard-threshold operator.

2.5.2 Alternating minimization

We introduce a widely used non-convex optimization primitive namely the alternating minimization principle. The technique is extremely general and its popular use actually predates the recent advances in non-convex optimization by several decades. Indeed, the popular Lloyd's algorithm [Llo82] for k-mean clustering and the EM algorithm [DLR77] for latent variable models, are problem specific variants of the general alternating minimization principle.

Consider the optimization problem:

$$\min_{x,y} \Psi(x,y) := f(x) + g(y) + \mathrm{H}(x,y),$$

where the functions f and g are extended values (i.e, allowing the inclusion of constraints) and H is a smooth function. The standard approach to solve this problem is via the so-called Gauss-Seidel iteration scheme, popularized in modern era under the name alternating minimization. That is, starting with some given initial point $(x^{(0)}, y^{(0)})$, we generate a sequence $\{(x^{(k)}, y^{(k)})\}_{k\in\mathbb{N}}$ via the scheme

$$\begin{aligned} x^{(k+1)} &\in \operatorname*{argmin}_x \Psi(x, y^k) \\ y^{(k+1)} &\in \operatorname*{argmin}_y \Psi(x^{k+1}, y). \end{aligned}$$

Convergence results for the Gauss-Seidel method can be found in several studies, see e.g [Aus71, BT03, Nes13, Tse01]. Otherwise, as shown in [Pow73], the method may cycle indefinitely without converging.

In the convex setting, for a continuously differentiable function Ψ, assuming strict convexity of one argument while the other is fixed, every limit point of the sequence $\{(x^{(k)}, y^{(k)})\}_{k\in\mathbb{N}}$ generated by this method minimizes Ψ, [BT03].

Removing the strong convexity assumption can be achieved by coupling the method with a proximal regularization of the Gauss-Seidel scheme [SNW12]:

$$\begin{aligned} x^{(k+1)} &\in \operatorname*{argmin}_x \{\Psi(x, y^k) + \frac{c_k}{2}||x - x^k||^2\} \\ y^{(k+1)} &\in \operatorname*{argmin}_y \{\Psi(x^{k+1}, y) + \frac{d_k}{2}||y - y^k||^2\}. \end{aligned}$$

The generalized alternating minimization algorithm is given as follows:

Algorithme 2.3 : Alternating minimization algorithm.

Input : Guess $(x^{(0)}, y^{(0)})$, Objective function $\Psi(x,y)$
Output : A point $(x^\star, y^\star)$ with near optimal value
for $k = t, \ldots, \mathrm{T}$ **do**
  $x^{(t+1)} \longleftarrow \operatorname*{argmin}_x \Psi(x, y^t)$
  $y^{t+1} \longleftarrow \operatorname*{argmin}_y \Psi(x^{t+1}, y)$

We note that the choice of two blocks of variables is for the sake of simplicity of exposition. Indeed, all results hold true for a finite number of block-variables.

2.6 Linear programming

Linear programming (LP) is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships [Wik20]. Linear programming is a special case of mathematical programming (also known as mathematical optimization). More formally, linear programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints.

The general linear programming problem is z_{LP}:

$$\begin{array}{ll} \max\limits_{x\in\mathbb{R}^n} & c^{\text{T}}x \\ \text{s.t} & \text{A}x \quad \leq b, \end{array}$$

where A is an $m \times n$ matrix, c is a $n \times 1$ matrix and b is an $m \times 1$ matrix [WN99]. We do not consider equality constraints since it can be shown that an equality constraint can be represented by two inequality constraints.

A linear programming (LP) is well-defined in the sense that if it is feasible and does not have unbounded optimal value, then it has an optimal solution.

Linear programming problems can be solved using the simplex algorithms which are part of linear programming software systems. Their performance shows that they are very robust and efficient.

Definition 2.6.1 (Duality) *Duality deals with pairs of linear programs and the relationships between their solutions. One problem is called the primal and the other the dual. We state the dual problem of z_{LP} as w_{LP}:*

$$\begin{array}{ll} \max\limits_{u\in\mathbb{R}^m} & b^{\text{T}}u \\ s.t & \text{A}^{\text{T}}u \quad \geq c. \end{array}$$

It does not matter which problem is called the dual because the dual of the dual is the primal.

Proposition 2.6.1 (Weak duality) *If $x^\star$ is primal feasible and $u^\star$ is dual feasible then:*

$$c^{\text{T}}x^\star \leq z_{\text{LP}} \leq w_{\text{LP}} \leq b^{\text{T}}u^\star.$$

This means that feasible solutions to the dual problem provide upper bounds on Z_{LP}, and feasible solutions to the primal problem yield lower bounds on w_{LP}.

Theorem 2.6.1 (Strong duality) *If z_{LP} or w_{LP} is finite, then both primal and dual have finite optimal value $z_{\text{LP}} = w_{\text{LP}}$.*

For more details about linear programming and proofs see for instance the Nocedal and Wright's reference book [NW06].

2.7 Mixed Integer Programming

A mixed integer program (MIP) can be written in the following format:

$$\min\{e^{\text{T}}x + h^{\text{T}}y : \text{F}x + \text{G}y \leq b, x \in \mathbb{Z}_+^n, y \in \mathbb{R}^p\}, \tag{2.7}$$

where e and h are vectors of size n and p, respectively [WN99]. Matrices F and G have size $m \times n$ and $m \times p$, respectively. Vector b is a column vector of size m. We recall that if all variables are integer, then the problem is and integer program. If all variables are $0-1$, then then the problem is a binary integer program or a $0-1$ integer program. Let $z := (x, y)$, $q = n + p$, $\mathrm{A} = (\mathrm{F}, \mathrm{G})$, $c_1 = (e, h)$ and $\mathrm{S} := \{z \in \mathbb{Z}_+^n \times \mathbb{R}^p : \mathrm{A}x \leq b\}$.
An optimization problem $min\{c_2^{\mathrm{T}} z : z \in \mathrm{T}\}$ where $c_2 \in \mathbb{R}^q$ is a relaxation of the problem (2.7) if $\mathrm{S} \subseteq \mathrm{T}$ and $c_2^{\mathrm{T}} z \leq c_1^{\mathrm{T}} z$ for all $z \in \mathrm{S}$. One widely used relaxation is the so called linear programming (LP) relaxation where the integrality of variables are relaxed in the MIP problem (2.7), i.e. if we replace $x \in \mathbb{Z}_+^n$ by $x \in \mathbb{R}_+^n$ then we obtain the LP relaxation of (2.7). Although in some special cases MIP problems can be polynomially solvable, in general they are known to be $\mathscr{NP}$-complete. In other words, unless $\mathscr{P} = \mathscr{NP}$ most MIP problems cannot be solved in polynomial time. However, in practice the branch-and cut algorithm is very successful in solving MIP problems. This algorithm combines the advantages of the cutting plane and the branch-and-bound algorithms. The LP relaxation of a problem is used as a starting point for all three algorithms. Furthermore, all the algorithms repeatedly solve LP problems while introducing new constraints. LP problems are much easier to solve compared to solving MIP problems.

2.7.1 Branch and Bound Algorithm

The branch-and-bound method was developed by Land and Doig [LD10] in 1960. To solve a MIP problem with this method, one initially solves the LP relaxation of the given MIP problem. According to the solution of the LP relaxation, the method starts its branching process. For a given solution $\hat{z}$, if $\hat{z} \notin \mathbb{Z}_+^n \times \mathbb{R}^p$, then the method creates two nodes that divide the mixed-integer feasible set with constraints $z_i \leq [\hat{z}_i]$ and $z_i \geq [\hat{z}_i]$, $i = 1, \ldots, n$. A node is clipped under three conditions: (i) a feasible solution is found, (ii) problem is infeasible and (iii) the lower bound obtained from the node exceeds the best available upper bound. If a node is not pruned then new nodes are created according to the fractional solution at the current active node and the branching continues. The algorithm ends when all the nodes are explored.

2.7.2 Cutting Plane Algorithm

A cutting plane algorithm is similar to the branch-and-bound algorithm in that it starts with solving the LP relaxation of the given MIP problem and repeatedly solves LP problems until it finds the optimal solution. However, instead of branching and creating nodes at each step the algorithm adds valid inequalities to cut off infeasible solutions. A valid inequality for a given MIP problem (2.7) is an inequality in the form of $\pi z \leq \pi_0$ and is satisfied by all $z \in \mathrm{S}$. Before using this algorithm, one needs to find valid inequalities to add to the LP relaxation. Finding a good class of valid inequalities that take into consideration the structure of the specific problem at hand is more of an art than a scientific procedure that has defined steps. Once a class of valid inequalities is available the next step is to find the most violated inequality (if any exists). This is called the separation problem. If a class of valid inequalities is adequate to find the optimal solution of set S for any objective function vector $c \in \mathbb{R}^q$, then this class is enough to describe the convex hull of S (i.e., $conv(\mathrm{S})$). The valid inequalities that make up the convex hull are called facet-defining inequalities of $conv(\mathrm{S})$. However, for most MIP problems, it is not possible to find the valid inequalities that define the convex hull.

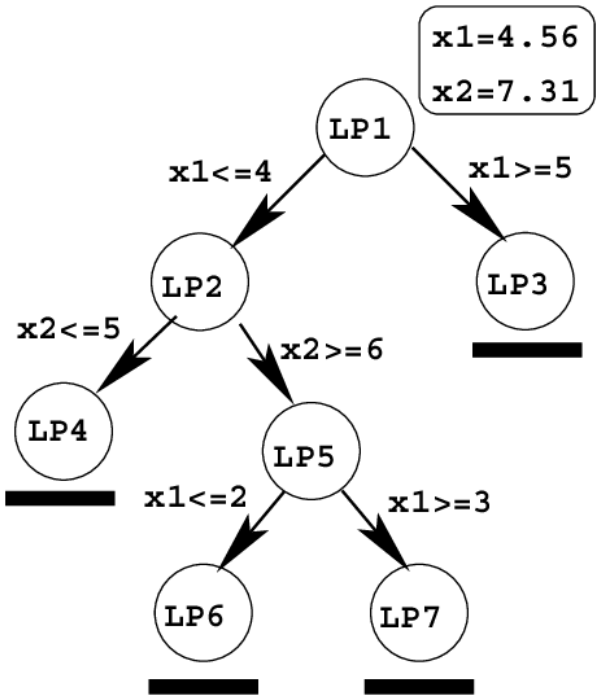

FIGURE 2.5 – A sketch of the working principle of the branch-and-bound method for a twovariable integer program. Every LP having a non-integer solution is branched into two LPs by adding an extra constraint on the variable. [DP04].

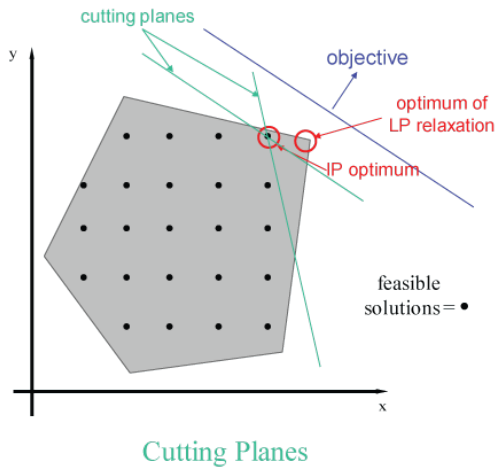

FIGURE 2.6 – One Simple Example of a Cutting Plane (from `gurobi.com/resource/mip-basics/`).

2.7.3 Branch and Cut Algorithm

The branch-and-cut algorithm is a hybrid of the branch-and-bound and the cutting plane algorithms. This hybrid method is widely used in the literature to solve MIP problems. Both the branch-and-bound and the cutting plane algorithms have drawbacks. For example, the branch-and-bound algorithm might create a large branchand-bound tree and the solution time can suffer due to solving too many LP problems. Similarly, the cutting plane algorithm will most likely add exponentially many valid inequalities and hence slow down the algorithm. In the branch-and-cut algorithm the cutting plane algorithm is called throughout the branch-and-bound tree. One can decide to call the cutting plane algorithm at every node or for example for the first c many nodes.

2.7.4 Mixed Integer Quadratic Program

Throughout more than 50 years of existence, mixed integer linear programming (MILP) theory and practice have been significantly developed, and it is now an indispensable tool in business and engineering. Two reasons for the success of MIP are the actual efficiency of linear programming (LP) based solvers and the modeling flexibility of MIP. We now have several extremely effective state-of-the-art solvers that incorporate many advanced techniques and, since its early stages, MIP has been used to model a wide range of applications. While all state of the art MIP solvers are based on the branch-and-bound algorithm, they also include a large number of advanced techniques that make it hard to predict the specific impact of an alternative formulation. However, there are two aspects of an MIP formulation that usually have a strong impact on both simple branch-and-bound algorithms and state of the art solvers: the size and strength of the LP relaxation, and the effect of branching on the formulation.
We are interested in mixed integer quadratic optimization problem (MIQP) that is a probelm of the form:

$$\begin{cases} \min & \frac{1}{2}\alpha^{\mathrm{T}} Q\alpha + \alpha^{\mathrm{T}} a \\ \text{s.t.} & \mathrm{A}\alpha \leq b \\ & \alpha_i \in \{0,1\}, \ \ \forall i \in \mathrm{I} \\ & \alpha_j \in \mathbb{R}, \ \ \forall j \notin \mathrm{I}, \end{cases} \tag{2.8}$$

where $a \in \mathbb{R}^m$, $\mathrm{A} \in \mathbb{R}^{k\times m}$, $b \in \mathbb{R}^k$ and $\mathrm{Q} \in \mathbb{R}^{m\times m}$ are the given parameters of the problem, $\leq$ denotes element-wise inequalities and we optimize over $\alpha \in \mathbb{R}^m$ containing both discrete ($\alpha_i, i \in \mathrm{I}$) and continuous ($\alpha_i, i \notin \mathrm{I}$) variables, with $\mathrm{I} \subseteq \{1,\ldots,m\}$. For background on MIP see [BW05]. Modern integer optimization solvers such as Gurobi and Cplex are able to tackle MIQO problems.

In the last twenty-five years (1991-2014) the computational power of MIP solvers has increased at an astonishing rate. In [Bix12], to measure the speedup of MIP solvers, the same set of MIP problems were tested on the same computers using twelve consecutive versions of Cplex and version-on-version speedups were reported. The versions tested ranged from Cplex 1.2, released in 1991 to Cplex 11, released in 2007, see Figure (2.7). Each version released in these years produced a speed improvement on the previous version, leading to a total speedup factor of more than 29,000 between the first and last version tested. Gurobi 1.0, a MIO solver which was first released in 2009, was measured to have similar performance to Cplex 11. Version-on-version speed comparisons of successive Gurobi releases have shown a speedup factor of more than 20 between Gurobi 5.5, released in 2013, and Gurobi. The combined machine-independent speedup factor in MIO solvers between 1991 and 2013 is 580,000. This impressive speedup factor is due to incor-

porating both theoretical and practical advances into MIO solvers. Cutting plane theory, disjunctive programming for branching rules, improved heuristic methods, techniques for preprocessing MIPs, using linear optimization as a black box to be called by MIP solvers, and improved linear optimization methods have all contributed greatly to the speed improvements in MIP solvers [Bix12]. For more details about the improvement of the MIP solvers, see [Lin17].

In addition, the past twenty years have also brought dramatic improvements in hardware. The hardware speedup from 1993 to 2013 is approximately $10^{5.5}$ 320000. When both hardware and software improvements are considered, the overall speedup is approximately 200 billion! Note that the speedup factors cited here refer to mixed integer linear optimization problems, not MIQP problems. The speedup factors for MIQP problems are similar. MIP solvers provide both feasible solutions as well as lower bounds to the optimal value. As the MIP solver progresses towards the optimal solution, the lower bounds improve and provide an increasingly better guarantee of suboptimality, which is especially useful if the MIP solver is stopped before reaching the global optimum. In contrast, heuristic methods do not provide such a certificate of suboptimality.

The belief that MIP approaches to problems in statistics are not practically relevant was formed in the 1970s and 1980s and it was at the time justified. Given the astonishing speedup of MIP solvers and computer hardware in the last twenty-five years, the mindset of MIP as theoretically elegant but practically irrelevant is no longer justified. In this paper, we provide empirical evidence of this fact in the context of the problem of feature selection and outlier detection in linear regression and support vector machines.

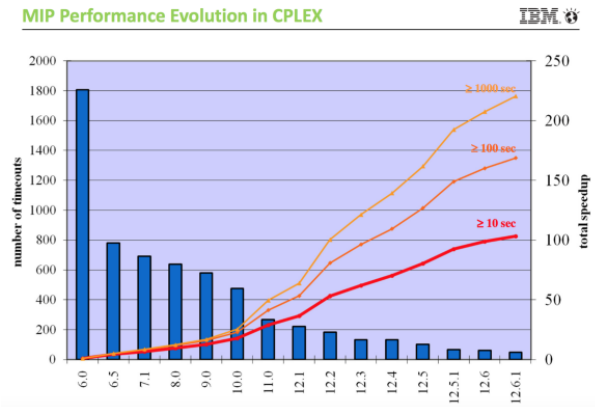

FIGURE 2.7 – Mixed Integer Programming Evolution in Cplex [Lin17].

2.8 Conclusion

This chapter introduced a mathematical background on optimization tools used in this work. We presented different convex constrained and unconstrained optimization problems together with some methods and algorithms to solve these problems. In addition, non-convex optimization algorithms were introduced, and alternating minimization algorithm which is used to solve problems with two or more blocks of variables was also presented. Linear Programming and Integer Programming were briefly introduced, while a fast survey on solving Mixed Integer Programming problems was presented. In general, all mathematical tools needed in the rest of the thesis were introduced.

Chapitre 3

Robust and Sparse Linear Regression

Sommaire

3.1 Introduction

Linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression [Fre09]. Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications [YS09]. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

— If the goal is prediction, forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response,

— If the goal is to explain variation in the response variable that can be attributed to variation in the explanatory variables, linear regression analysis can be applied to quantify the strength of the relationship between the response and the explanatory variables, and in particular to determine whether some explanatory variables may have no linear relationship with the response at all, or to identify which subsets of explanatory variables may contain redundant information about the response.

3.2 Least squares

Linear regression models are often fitted using the least squares approach. We consider the linear regression model with response vector $y_{n\times 1}$, model matrix $\mathrm{X} = [x_1, \ldots, x_p] \in \mathbb{R}^{n\times p}$, regression coefficients $\beta \in \mathbb{R}^{p\times 1}$ and errors $\epsilon \in \mathbb{R}^{n\times 1}$:

$$y = \mathrm{X}\beta + \epsilon.$$

Here β is the vector of unknown coefficients, and the variables x_j can come from different sources:

— quantitative inputs;

— transformations of quantitative inputs, such as log, square-root or square;

No matter the source of the x_j, the model is linear in parameter [FHT01]. To estimate β, the most popular estimation method is the *least squares* in which we pick the coefficients β to minimize the residual sum of squares:

$$\begin{aligned} \mathrm{RSS}(\beta) &= ||y - \mathrm{X}\beta||_2^2 \\ &= \sum_{i=1}^{n} \big(y_i - \sum_{j=1}^{p} \mathrm{X}_{ij}\beta_j\big)^2. \end{aligned} \tag{3.1}$$

The objective is to minimize (3.1) which is a quadratic function in β. Differentiating with respect to β we obtain:

$$\begin{aligned} \frac{\partial \mathrm{RSS}}{\partial \beta} &= -2\mathrm{X}^{\mathrm{T}}(y - \mathrm{X}\beta) \\ \frac{\partial^2 \mathrm{RSS}}{\partial \beta \partial \beta^{\mathrm{T}}} &= 2\mathrm{X}^{\mathrm{T}}\mathrm{X}. \end{aligned} \tag{3.2}$$

Assuming (for the moment) that X has full column rank, and hence $\mathrm{X}^{\mathrm{T}}\mathrm{X}$ is positive definite, we deduce that (3.1) is convex. Therefore, to minimize it, it is sufficient to set its first derivative to zero, that is:

$$\mathrm{X}^{\mathrm{T}}(y - \mathrm{X}\beta) = 0,$$

we can easily deduce that if $\mathrm{X}^{\mathrm{T}}\mathrm{X}$ is invertible i.e $rank(\mathrm{X}) \leq n$, we have:

$$\hat{\beta} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}} y,$$

hence the predicted values at x are given by:

$$\hat{y} = x^{\mathrm{T}}\hat{\beta} = x^{\mathrm{T}}(\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}} y.$$

For more details see for instance [Fre09].

Overfitting

Overfitting refers to a model that models the training data too well. It happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

Bias-Variance trade-off

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.
While variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it has not seen before. As a result, such models perform very well on training data but has high error rates on test data.

Mathematically, let the variable we are trying to predict as y and other covariates as X. We assume there is a relationship between the two such that:

$$y = f(\mathrm{X}) + \epsilon,$$

where ϵ is the noise term of variance σ_ϵ^2 and it is normally distributed with a mean of zero. Let $\hat{f}(\mathrm{X})$ be the model estimator using linear regression. Then the expected squared error at a point x is:

$$\begin{aligned} Err(x) &= \mathrm{E}\left[(y - \hat{f}(x))^2\right] \\ &= \underbrace{\left(\mathrm{E}[\hat{f}(x)] - f(x)\right)^2}_{\text{Bias}^2} + \underbrace{\mathrm{E}\left[\left(\hat{f}(x) - \mathrm{E}[\hat{f}(x)]\right)^2\right]}_{\text{variance}} + \underbrace{\sigma_\epsilon^2}_{\text{Irreducible error}}. \end{aligned}$$

where the irreducible error is the error that can't be reduced by creating good models. It is a measure of the amount of noise in our data. It is also important to understand that no matter how good we make our model, our data will have certain amount of noise or irreducible error that can not be removed.
Underfitting happens when a model is unable to capture the underlying pattern of the data. These models usually have high bias and low variance. While overfitting happens when our model captures the noise along with the underlying pattern in data. These models usually have high variance and low bias. To this end, we need to find the right/good balance without overfitting and underfitting the data. This is called bias-variance trade-off.Thus to build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.

Ridge regression

In the high dimensional case, when $p > n$, the ordinary least squares estimator is not unique and will heavily overfit the data. Thus, a form of complexity regularization will be necessary [BVDG11]. Moreover, in high dimensional case, with a large number of predictors, it can be helpful to identify a smaller subset of important variables to ensure a good

interpretation performance. To overcome the problem of overfitting, the ridge regression estimator was proposed by [HK70]. In ridge regression, the cost function is altered by a penalty equivalent to square of the magnitude of the coefficients, that is:

$$\min_{\beta \in \mathbb{R}} \quad ||X\beta - y||_2^2 + \lambda ||\beta||_2^2. \tag{3.3}$$

The problem (3.3) can be shown equivalent to the following optimization problem:

$$\begin{cases} \min\limits_{\beta \in \mathbb{R}} & ||X\beta - y||_2^2 \\ & ||\beta||_2^2 \leq c, \end{cases} \tag{3.4}$$

that is, a one-to-one correspondence exits between λ in (3.3) and c in (3.4). In (3.3) $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage [Fre09], we note that:

— when $\lambda = 0$, we get the linear regression estimate;

— when $\lambda = +\infty$, we get $\beta^{\text{ridge}} = 0$;

— For λ in between, we are balancing two ideas: fitting a linear model of y on X, and shrinking the coefficients;

The bias and variance are not quite as simple to write down for ridge regression as they were for linear regression but closed-form expressions are still possible. The general trend is :

— The bias increases as λ (amount of shrinkage) increases;

— The variance decreases as λ (amount of shrinkage) increases;

We recall that the solution of the ordinary least squares problem (3.1) is given by:

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

In the high dimensional case, $p \gg n$, the matrix $X^T X$ is not of full rank, and thus it is not invertible and the solutions $\hat{\beta}$ is no more unique. While it is easy to verify that the solution of (3.3) is given by:

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y,$$

where I is the $p \times p$ identity matrix, we can conclude that the ridge regression methods adds a positive constant to the diagonal of $X^T X$ before inversion, which makes the problem non-singular even if $X^T X$ is not of full rank. This was the main motivation for ridge regression when it was first introduced in statistics [HK70]. To see why this is true, first remark that $X^T X$ is a $p \times p$ symmetric matrix, hence its eigen decomposition is given by: $X^T X = VDV^T$ where

$$D = \begin{pmatrix} d_1 & 0 & 0 & \cdots & 0 \\ 0 & d_2 & 0 & \cdots & 0 \\ 0 & 0 & d_3 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & d_p \end{pmatrix}$$

with $d_i \geq 0$. Now since matrix inversion corresponds to the inversion of of the eigenvalues, then $(X^T X)^{-1} = VD^{-1}V^T$ (note that $V^T = V^{-1}$), this works if all eigenvalues are strictly greater than zero, that is $d_i > 0 \quad i = 1, \ldots, p$. However, for $p \gg n$ this is impossible, moreover a

TABLE 3.1 – Examples of convex penalty functions [AGN11].

Convex	
Smooth at zero	Singular at zero
$P(\beta) = \lvert\beta\rvert^{\alpha}, \alpha > 1$	$P(\beta) = \lvert\beta\rvert$ $P'(0^+) = 1$
$P(\beta) = \sqrt{\alpha + \beta^2}$	$P(\beta) = \alpha^2 - (\lvert\beta\rvert - \alpha)^2 I\{\lvert\beta\rvert < \alpha\}$ $P'(0^+) = 2\alpha$
$P(\beta) = log(cosh(\alpha\beta))$	
$P(\beta) = \beta^2 - (\lvert\beta\rvert - \alpha)^2 I\{\lvert\beta\rvert > \alpha\}$	
$P(\beta) = 1 + \lvert\beta\rvert/\alpha - log(1 + \lvert\beta\rvert/\alpha)$	

small change or perturbation in any d_i will lead to a huge variation $\frac{1}{d_i}$ if d_i is very small. So what ridge regression does, is moving all eigenvalues further away from zero as:

$$\begin{aligned} X^T X + \lambda I_p &= V D V^T + \lambda I_p \\ &= V D V^T + \lambda V V^T \\ &= V(D + \lambda I_p) V^T, \end{aligned}$$

which now has eigenvalues $d_i + \lambda \geq \lambda > 0$. This is why choosing positive penalty parameter makes the matrix invertible even in the high dimensional case $p >> n$. [Fre09] present important references about the ridge regression.
Finally we will mention two drawbacks of the ridge regression:

— we should solve a $p \times p$ linear system, which implies $O(p^3)$ computation, which is hard when p is large;

— the solution $\hat{\beta}_{ridge}$ is dense, that is the majority of its components is not equal to zero, so the interpretation of the results will be difficult.

To this end, when p is large the notion of sparsity is always needed, where a sparse vector refers to a vector with few non-zero element.

Penalized Regression

Penalized regression methods keep all the predictor variables in the model but constrain (regularize) the regression coefficients by shrinking them toward zero. If the amount of shrinkage is large enough, these methods can also perform variable selection by shrinking some coefficients to zero.

The following equation shows the general form of the shrinkage and regularization methods for linear models:

$$\tilde{\beta} = arg \min_{\beta} ||y - X\beta||_2^2 \text{ s.t } P(\beta) \leq t,$$

where $P(\beta)$ is a penalty on the regression coefficients t is a positive tuning parameter.

The shrinkage (tuning) parameter t determines the amount of shrinkage on the regression coefficients. Note that if you choose t to be very large, you do not place a penalty on the size of the regression coefficients and thus the optimum $\tilde{\beta}$ is the OLS solution. As t decreases, regression coefficients shrink from the OLS solution toward zero.

In the last decade, many different penalized regression methods have been proposed. The LASSO [Tib96], adaptive lasso [Zou06] and elastic net [ZH05] are the most popular. Other penalties are also proposed and for each method, the penalty $P(\beta) \leq t$ imposed on the regression coefficients takes a different form as shown in Tables (3.1) and (3.2):

TABLE 3.2 – Examples of non-convex penalty functions [AGN11].

Non-Convex	
Smooth at zero	Singular at zero
$P(\beta) = \alpha\beta^2/(1+\alpha\beta^2)$	$P(\beta) = \lvert\beta\rvert^{\alpha},\ \alpha \in]0,1[\ \ P^{'}(0^{+}) = \infty$
$P(\beta) = min\{\alpha\beta^2, 1\}$	$P(\beta) = log(\alpha\lvert\beta\rvert + 1)\ \ P^{'}(0^{+}) = \alpha$
$P(\beta) = 1 - \exp -\alpha\beta^2$	
$P(\beta) = -log(\exp -\alpha\beta^2 + 1)$	

3.3 Feature selection

Feature selection reduces the dimensionality of data by selecting only a subset of measured features to create a model. Subset of features is evaluated using a criteria that usually involves minimization of a specific measure of predictive error. Feature Selection algorithms search for a subset of predictors that optimally model the response variable, subject to constraints such as required or excluded features and the size of the subset. Feature selection is preferable to feature transformation when the original units and meaning of features are important and the modeling goal is to identify an influential subset. When categorical features are present, and numerical transformations are inappropriate, feature selection becomes the primary means of dimensionality reduction.

It might seem that adding more variables to the predictive model can make the model utilizing more information ; thus, making it more reliable. However, the more variables we include into the model, the more flexible it becomes, and so we increase the overall probability for the model to overfit [Bis06]. So the final goal of Feature Selection is to keep the number of parameters used in the model as low as possible while keeping its predictive performance on an acceptable level. This way we reduce the probability of overfitting the model. There is a number of other potential benefits from Feature Selection: simplification of data visualization, decrease in storage requirements, reduction of training time, defying the curse of dimensionality to improve the overall prediction performance.

Ridge regression will never sets coefficients to zero exactly, and therefore cannot perform variable selection in the linear model. While this did not seem to hurt its prediction ability, it is not desirable for the purposes of interpretation (especially if the number of variables p is large). In order to conduct statistically meaningful inference, it is desirable to assume that the true regression coefficient β is sparse or may be well approximated by a sparse vector. A natural way to compute sparse regression coefficients is to solve the, well known, best subset selection problem:

$$\begin{cases} \min\limits_{\beta\in\mathbb{R}^p} & \frac{1}{2}\|X\beta - y\|_2^2 \\ \text{s.t.} & \|\beta\|_0 \leq k_v, \end{cases} \tag{3.5}$$

where the ℓ_0 norm of a vector β counts the number of nonzeros in β. This classical problem dates back to at least [BKM67, HL67]. It has been considered as intractable since it is an NP-hard problem [Nat95, Mil02]. Indeed, state-of-the-art algorithms to solve Problem (3.5), as implemented in popular statistical packages, like leaps in R, do not scale to problem sizes larger than $p = 30$. Due to this reason, it is not surprising that the best subset problem has been widely dismissed as being intractable by the greater statistical community. Feature Selection is solved by a number of different approaches [GE03] that can be divided into the following three categories:

— **Ranking the features.** A number of ranking statistical criteria has been developed that measures the relative usefulness of every feature for making predictions. After

ranking, we can select the best k features and build a model using only the selected attributes.
Formally speaking, we have a data set of n observations $(x_i, y_i)_{i=1}^n$ each consisting of p independent variables $x_i \in \mathbb{R}^p$ and one output dependent variable y_i. Let x_{ik} be the kth component of the vector x_i. We would like to select at most $\lambda \leq p$ highly relevant features. We use a scoring function $S(k) \in \mathbb{R}$ for $k = 1 \dots p$ computed from the kth feature x_{ik} and y_i for every $i = 1, \dots, n$. We will assume that a high score indicates high relevancy of the variable in the sense of the used scoring function. So after calculating $S(k)$ for every $k = 1 \dots d$ we sort all features in the decreasing order of $S(k)$ and select the top λ independent variables. There are other techniques for selecting an optimal subset of features [BBE+03, SDDO03]. The main idea of these techniques is to introduce a random probe in the data by adding artificial independent variables to the original data set that consist purely of gaussian noise. After sorting the features, we disregard independent variables that have lower relevancy than the newly constructed random variables since they potentially would provide less information for a predictive model than regular random noise. The most well-known and widely used scoring functions are Bayesian Information Criterion (BIC), Akaike information criterion (AIC), Correlation (R) and Coefficient of Determination (R^2) [Bis06]. However, the major drawback of ranking the features is that each criterion can suggest a different list of the best attributes.

— **Wrappers.** Methods of this type use a predictive machine learning algorithm as a perfect black box to evaluate the quality of a subset of attributes. To assess relevancy of a subset of the features, we need to select a predictive tool first, like a regression model or a classification algorithm, then we compute predictive performance of the selected model that uses only the subset of selected features. Next, if we are not satisfied with the quality of prediction, we modify the subset by following a predefined search strategy. We repeat this procedure until we get acceptable predictive performance or until we reach some stop criterion, like a maximum number of iterations or inability to modify the subset of features in two consecutive iterations. A wide range of search strategies can be used for selecting the best subset of features, among them are: sequential search, simulated annealing, genetic algorithms, etc. Actually, any general purpose heuristic can be used as a search strategy. Using a learning machine algorithm as a black box, wrappers are universal and simple to implement. Moreover, wrappers not only result in a good subset of relevant features but they also produce a trained predictive model. These methods are believed to be the most accurate; however, some criticize them as simple "brute force" methods since they require a massive amount of computation [RPD01]. One of the most common search strategies that is used in conjunction with linear regression is sequential search. This type of regression called Stepwise linear regression. There are 3 variations of sequential search:

1. **Forward selection.** The regression is initialized with only one feature added into the model that is the most correlated with the response variable. Then it searches for a feature that reduces a mean squared or absolute error the most. If this decrease of the mean error is greater than some threshold, we continue adding new features until new independent variable does not decrease the mean error for more than the threshold value.
2. **Backward elimination.** This variation of regression starts with all the independent variables added into the model. Then it disregards one feature at a time that does not increase mean error for more than the threshold. We stop

when we can not remove any additional feature from the subset of chosen features without increasing the predictive error for too much. This process is opposite to Forward selection. Unfortunately, we can not use this method alone, if the number of observations we have is less than the number of features.

3. **Bidirectional search.** This method is a combination of the above two. We start with only one feature correlated with the response variable, then we apply Forward selection until no more additional features can be added into the model. After that the Backward elimination procedure starts, until no more features can be removed from the model without increasing the mean error too much. We continue repeating these 2 search heuristics, until we reach a subset of independent variables that can not be alternated by any of the two procedures.

— **Embedded Methods.** Embedded methods are machine learning and statistical algorithms that have a built-in Feature Selection. Usually they include a regularization term that penalizes the objective function of the method if it adds too many features into the model (like in LASSO [Tib96], Multivariate adaptive regression splines (MARS) [Fri91] or Random multinomial logistic methods [FHT10]) or any method that uses pruning as an intermediate step in the model building process for making the model less complex, like in Decision Trees [GE03].

LASSO

To overcome the computational difficulties of the best subset selection problem, computationally tractable convex optimization based methods like Lasso [Tib96] have been proposed as a convex surrogate for Problem (3.5). For the linear regression problem, the lasso problem is given by:

$$\begin{cases} \min\limits_{\beta\in\mathbb{R}^p} & \frac{1}{2}\|X\beta - y\|_2^2 \\ \text{s.t.} & \|\beta\|_1 \leq t, \end{cases} \tag{3.6}$$

where $||\beta||_1 = \sum_{i=1}^{p} |\beta_i|$. Problem (3.6) is the constrained form of the lasso regression model. There has been a lot of impressive work on lasso [EHJ+04, CP+09, BRT+09, ZH+08, ZY06, Tib11] in terms of algorithms and understanding of its theoretical properties. Lasso enjoys several attractive statistical properties and has drawn a significant amount of attention from the statistics community.
The lasso can be written in the equivalent Lagrangian form [FHT01]:

$$\min_{\beta\in\mathbb{R}^p} \quad \frac{1}{2}||X\beta - y||_2^2 + \lambda||\beta||_1. \tag{3.7}$$

It can be shown that Problems (3.6) and (3.7) are equivalent. We also note that Problem (3.7) is a convex quadratic optimization problem and can be solved efficiently.

How to find lasso solutions?

We will show how problem (3.7) can be solved using two methods: coordinate descent and proximal gradient descent.

Coordinate descent:

$$\frac{1}{2}||y-X\beta||_2^2+\lambda||\beta||_1 \quad = \quad \frac{1}{2}\underbrace{||y-X\beta||_2^2}_{g}+\underbrace{\lambda\sum_{i=1}^{p}|\beta|}_{h}.$$

We remark that $g=\frac{1}{2}\sum_{i=1}^{n}\left[y^{(i)}-\sum_{j=1}^{p}\beta_j X_j^{(i)}\right]^2$, then differentiating g with respect to β_j leads to:

$$\begin{aligned}\frac{\partial g}{\partial \beta_j} &= -\sum_{i=1}^{n}X_j^{(i)}\left[y^{(i)}-\sum_{j=1}^{p}\beta_j X_j^{(i)}\right]\\ &= -\sum_{i=1}^{n}X_j^{(i)}\left[y^{(i)}-\sum_{k=1,k\neq j}^{p}\beta_k X_k^{(i)}-\beta_j X_j^{(i)}\right]\\ &= -\sum_{i=1}^{n}X_j^{(i)}\left[y^{(i)}-\sum_{k=1,k\neq j}^{p}\beta_k X_k^{(i)}\right]+\beta_j\sum_{i=1}^{n}\left(X_j^{(i)}\right)^2\\ &= -\rho_j+\beta_j z_j,\end{aligned}$$

with $\rho_j=\sum_{i=1}^{n}X_j^{(i)}\left[y^{(i)}-\sum_{k=1,k\neq j}^{p}\beta_k X_k^{(i)}\right]$ and $z_j=\sum_{i=1}^{n}\left(X_j^{(i)}\right)^2$.
h is not differentiable at 0, we can compute its subgradient, that is:

$$\frac{\partial h}{\partial \beta_j}=\begin{cases}-\lambda & \text{if} \quad \beta_j<0\\ [-\lambda,\lambda] & \text{if} \quad \beta_j=0\\ \lambda & \text{if} \quad \beta_j>0.\end{cases}$$

To find the minimum, we have to solve $0\in\frac{\partial g}{\partial \beta_j}+\frac{\partial h}{\partial \beta_j}$, that is:

$$0=\begin{cases}-\rho_j+\beta_j z_j-\lambda & \text{if} \quad \beta_j<0\\ [-\rho_j-\lambda,-\rho_j+\lambda] & \text{if} \quad \beta_j=0\\ -\rho_j+\beta_j z_j\lambda & \text{if} \quad \beta_j>0.\end{cases}$$

Then $0\in[-\rho_j-\lambda,-\rho_j+\lambda]\implies \quad -\lambda\leq\rho_j\leq\lambda$. Finally, we can deduce that the solution is given by:

$$\beta_j=\begin{cases}\frac{\rho_j+\lambda}{z_j} & \text{if}\ \rho_j<-\lambda\\ 0 & \text{if}\ -\lambda\leq\rho_j\leq\lambda\\ \frac{\rho_j-\lambda}{z_j} & \text{if}\ \rho_j>\lambda,\end{cases}$$

so that $\beta_j=\frac{1}{z_j}S(\rho_j,\lambda)$ where $S(\rho_j,\lambda)$ is the soft thresholding operator. We note that $z_j=1$ is the data is normalized.
The pseudo-code of the coordinate descent method for lasso is given by:

Algorithme 3.1 : Coordinate descent for lasso.

Input : data x and y
Output : β
for $j = 1 \dots p$
compute: ρ_j
Set: $\beta_j = S(\rho_j, \lambda)$

Proximal gradient method

For any given $y \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n \times p}$, the lasso criterion for linear regression is given by:

$$f(\beta) = \frac{1}{2} \underbrace{||y - X\beta||_2^2}_{g(\beta)} + \underbrace{\lambda ||\beta||_1}_{h(\beta)}.$$

The proximal operator for the lasso objective is computed as follows:

$$\begin{aligned} prox_t(\beta) &= arg\min_z \tfrac{1}{2t}||\beta - z||_2^2 + \lambda||z||_1 \\ &= arg\min_z \tfrac{1}{2}||\beta - z||_2^2 + \lambda t||z||_1 \\ &= S_{\lambda t}(\beta). \end{aligned}$$

where $S_\lambda(\beta)$ is the soft threshold operator given, for $i = 1, \dots, p$, by:

$$S_\lambda(\beta)_i = \begin{cases} \beta_i - \lambda & if \beta_i \geq \lambda \\ 0 & if\ |\beta_i| \leq \lambda \\ \beta_i + \lambda & if\ \beta_i \leq -\lambda. \end{cases}$$

Therefore, the proximal map for lasso objective is calculated by soft-thresholding β by amount λt. Next, we use the gradient of g, $\nabla g(\beta)$ which is same as the gradient of least squares function, i.e. $\nabla g(\beta) = -X^T(y - X\beta)$. From this, we obtain the following proximal gradient update:

$$\beta^+ = S_{\lambda t}(\beta + tX^T(y - X\beta)). \tag{3.8}$$

This simple algorithm calculates the lasso solution and is known as Iterative soft-thresholding algorithm (ISTA). It converges if $t \leq ||X^TX||$. We note that the proximal gradient method is faster than the coordinate descent method.

Lasso vs Ridge and Feature selection

Ridge regression is a regularized linear regression model. It enforces the β coefficients to be lower but it does not enforce them to be zero. That is, it will not get rid of irrelevant features but rather minimize their impact on the trained model.
Lasso is another extension built on regularized linear regression. The only difference from ridge regression is that the regularization term is in absolute value. This small difference has a huge impact. Indeed, lasso method overcomes the disadvantage of ridge regression by not only punishing high values of the coefficients β but actually setting them to zero if they are not relevant. Therefore, we might end up with fewer features included in the model than you started with, which is a huge advantage.

We also note that there exist many other variants of L_p penalties $||\beta||_p^p$, elastic net and Dantzig selector, see for instance [AADFG20].

Tuning parameter λ

The tuning parameter λ , sometimes called a penalty parameter, controls the strength of the penalty term in ridge regression and lasso regression. It is basically the amount of shrinkage, where data values are shrunk towards a central point, like the mean. We recall that when $\lambda = 0$, the estimate is equal to the one found with ordinary least squares. As λ increases more and more coefficients are set to zero and eliminated. As mentioned before, there is a trade-off between bias and variance in resulting estimators. As λ increases, bias increases and as λ decreases, variance increases. For example, setting your tuning parameter to a low value results in a more manageable number of model parameters and lower bias, but at the expense of a much larger variance.
Choosing the tuning parameter is a challenging task. Optimal tuning parameters are *difficult to calibrate in practice* [LM15]. However the tuning parameter λ is usually chosen via some cross-validation scheme aiming for prediction optimality [BVDG11].

Cross validation

Almost every machine learning algorithm comes with a large number of settings that we need to specify. These tuning knobs, the so-called hyperparameters, help us control the behavior of machine learning algorithms when optimizing for performance, finding the right balance between bias and variance. Hyperparameter tuning for performance optimization is an art in itself, and there are no hard-and-fast rules that guarantee best performance on a given dataset. In the machine learning community, the cross validation is the widely used method to tune these parameters.
Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. There are many cross validation strategies, however we are interested only in:

— hold-out Validation;

— k-fold cross validation;

Hold-out validation: To avoid over-fitting, an independent test set is preferred. A natural approach is to split the available data into two non-overlapped parts: one for training and the other for testing. The test data is held out and not looked at during training. Hold-out validation avoids the overlap between training data and test data, yielding a more accurate estimate for the generalization performance of the algorithm. The downside is that this procedure does not use all the available data and the results are highly dependent on the choice for the training/test split. The instances chosen for inclusion in the test set may be too easy or too difficult to classify and this can skew the results. Furthermore, the data in the test set may be valuable for training and if it is held-out prediction performance may suffer, again leading to skewed results. These problems can be partially addressed by repeating hold-out validation multiple times and averaging the results, but unless this repetition is performed in a systematic manner, some data may be included in the test set multiple times while others are not included at all, or conversely some data may always fall in the test set and never get a chance to contribute to the learning phase. To deal with these challenges and utilize the available data to the max, k-fold cross-validation is used.

k-fold cross validation: In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently k iterations of training and vali-

dation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining $k-1$ folds are used for learning. Data is commonly stratified prior to being split into k folds. Stratification is the process of rearranging the data as to ensure each fold is a good representative of the whole. $5-$ and $10-$ fold cross validation are mostly used.

3.4 Outlier detection

Outliers are understood to be observations that have been corrupted, incorrectly measured, mis-recorded, drawn under different conditions than those intended, or so atypical as to require separate modeling [YXW+10]. Dealing with the presence of outliers can be divided into two categories: (a) the so-called "robust statistics", that is robust-to-outlier loss functions and (b) outlier detection per se which excludes outliers from the training set.

Outliers may appear for many reasons, but the most common ones are experimental errors and a priori unknown variability in the measured properties, which we are trying to explain and predict. Usually outliers are excluded before applying statistical analysis; however, abnormal observations should be investigated separately from good observations since some of them may carry additional knowledge useful for our analysis and provide us with a key for deeper understanding of the studied phenomena [Coo77]. Many techniques exist that allow us to find potential outliers [HA04]. In general, they can be divided into two categories: statistical and machine learning techniques. Statistical approaches were the first methods of outlier detection. The basic idea behind the statistical methods is that all the observations are sampled from some, possibly unknown, distribution and so, outliers are the values that are very unlikely to occur from that distribution. One of the earliest approaches was Grubb's method (Extreme Studentized Deviate) [Gru69], which calculates a so called Z value of each observation. Z value is the difference between the mean of the response variable and the query value is divided by the standard deviation of the response variable, where the mean and standard deviation are calculated from all the observations including the query value. After computing the Z value is compared with 1% or 5% significance level. This method do not use any input parameters. However, it is sensitive to the number of observations and it assumes that the response variable is normally distributed, which is not always true. Another simple and well-known statistical tool used for potential outlier detection is a method called boxplot [LJK+00] that uses graphical plots for revealing the outliers in a given sample. The boxplot is a diagram that visually represents the distribution of the sample we study, highlighting where the majority of values lie as well as shows abnormal observations that lie too far from the majority of points.

As for the outlier detection techniques that are based on machine learning algorithms, they rely on the idea that good observations are grouped into some clusters, while abnormal observations lie far from the clusters. These methods rely on proximity information from the data, like different clustering techniques and especially methods that are based on k-Nearest Neighbours. Proximity based techniques make no prior assumptions about the data distribution and usually are very easy to implement. However, there are two drawbacks in such approaches. First, there must be a clear justification which distance metric we should use to calculate the proximity between any two observations. And second, machine learning algorithms are very susceptible to the curse of dimensionality. While the most effective statistical techniques automatically focus on the suspicious observations, the machine learning methods need to compute the distance between any two

points and so increase in the data dimensionality will lead to at least quadratic increase in time complexity. This is a common issue for the methods based on clustering or k-Nearest Neighbour algorithms. To overcome these problems, the data need to be preprocessed first to select the most relevant features and/or to reduce the number of observations leaving only the most representative [AB94, Ska94].

All these techniques are general-purpose and can be applied before any data analysis routines. However, if we need to perform regression specifically, we have another option. We can use regression techniques that are less sensitive to the presence of outliers. These regression techniques are called robust. The most widely used Least Sum of Squares estimator is a well-known non-robust lack-of-fit function. Even the presence of only one abnormal observation may change the behaviour of the ordinary least squares (OLS) regression [HRRS11]. On the other hand, another popular estimator, Least Sum of Absolute Deviations (LAD), is a known robust estimator since even the presence of several outliers does not change much its results comparing to results obtained on a data set without outliers. Many more robust estimators have been developed and are still being developed, like the general M-estimator [Mal75], Least Trimmed Squares [Rou84], S-estimator [RY84], MM-estimator [Yoh87] and their variations [ZA05] to account different assumptions, conditions and models. Nevertheless, despite of having many tools for detecting potential outliers, they still pose a problem for regression modelling. The reason is that no formal definition of an outlier exists, and so different techniques most likely will label different observations as outliers

3.5 Feature Selection and Outliers Detection Literature

The first attempt to solve the sparse robust regression issue was the LAD-Lasso, proposed by [WLJ07]. The idea was to use the least absolute deviation (LAD) loss to outliers together with the ℓ_1 Lasso penalty that is for a given regularization parameter $\lambda \geq 0$,

$$\min_{\beta} \|X\beta - y\|_1 + \lambda\|\beta\|_1.$$

To solve this problem, the idea was to boil it down to an augmented unpenalized LAD and use standard tools. Later, [LZ08] showed that this problem was a particular case of a more general class of parametric LP problems and exhibited its solution path according to λ. [WWL12] showed that this was a particular case of a general problem by considering generalization to check loss (associated with quantile regression) and non convex penalties such as SCAD and MCP. Almost the same idea was introduced by [WYG+08] in a totally different context motivated by the face recognition problem, in which sparse error appears due to a fraction of the query image being occluded by glasses , hats, etc. The novelty was the explicit introduction of a sparse variable τ modeling the corrupted pixels (aka the outliers) (a fraction of its entries are nonzero)

$$\begin{aligned} \min_{\beta,\tau} \quad & \|\beta\|_1 + \|\tau\|_1 \\ s.t. \quad & y = X\beta + \tau. \end{aligned}$$

In subsequent work, [WMM+10] acknowledged that this was the convex relaxation of the following problem

$$\begin{aligned} \min_{\beta,\tau} \quad & \|\beta\|_0 + \|\tau\|_0 \\ s.t. \quad & y = X\beta + \tau. \end{aligned}$$

It has also been pointed out as further work in [WM10] to demonstrate the stability of this model with respect to noise. It implies that $y = X\beta + \tau + \varepsilon$ where ε, can be seen as a dense bounded Gaussian noise. To achieve exact recovery from measurements corrupted with sparse noise [LDB09] proposed an algorithm under the name of justice pursuit denoising:

$$\begin{aligned} \min_{\beta,\tau} \quad & \|\beta\|_1 + \|\tau\|_1 \\ s.t. \quad & \|X\beta + \tau - y\|_2 \leq \sigma^2, \end{aligned}$$

[NT12] generalizes by introducing a controlled parameter $\lambda \geq 0$ that balances the two ℓ_1-norm terms

$$\begin{aligned} \min_{\beta,\tau} \quad & \|\beta\|_1 + \lambda\|\tau\|_1 \\ s.t. \quad & \|X\beta + \tau - y\|_2 \leq \sigma^2. \end{aligned}$$

equivalent to (since its a convex problem)

$$\min_{\beta,\tau} \|X\beta + \tau - y\|_2 + \mu\|\beta\|_1 + \lambda\|\tau\|_1.$$

this can be seen as a convexification of the brute force algorithm [CCM13]:

$$\begin{aligned} \min_{\beta,\tau} \quad & \|X\beta + \tau - y\|_2 \\ s.t. \quad & \|\beta\|_0 \leq k_v \text{ and } \|\tau\|_0 \leq k_o. \end{aligned}$$

With the argument that ℓ_0 brings the ideal solution but the associated optimization problem is NP Hard, [CCM13] replaced the inner product used by the thresholding regression, the lasso and the Dantzig selector, with a trimmed inner product. They obtained three robust algorithms: the robust thresholding regression, the robust lasso and the robust Dantzig selector.
[ACG+13] introduced the sparse least trimmed squares estimator. The idea was to use the least trimmed squares estimator as a robust estimator, and the lasso to ensure sparsity. [BJK15] proposed an algorithm called "Torrent: Thresholding Operator-based Robust Regression Method". Torrent is based on estimating an active set of points which have the least residual error on the current regressor, and then updating the regressor to provide a better fit on this active set. [ÖAC16] introduced the shooting S-estimator combining the ideas of the coordinate descent algorithm with simple S-regression. The introduced estimator is especially designed for situations where a large number of observations suffers from contamination in a small number of predictor variables. The thresholded justice pursuit [Des19] is also an interesting estimator in this area. Other robust versions of the lasso have been considered in the literature [DBJ+20], most of them are penalized M-estimators as in ([VdG+08]), ([LPZ11]) and ([DT19]).

In this chapter, we propose the use of the ℓ_0 norm as introduced by ([BKM16]), to perform both variable selection and outliers detection.

Brief Context and Background

Let $X = (x_1, \ldots, x_n)^t$ be a $n \times p$ design matrix and $y \in \mathbb{R}^n$ a response vector. We consider the following linear model to accommodate outliers:

$$\forall i \in \{1, \ldots, n\}, \qquad y_i = \begin{cases} x_i^t\beta + \varepsilon_i & \text{if observation } i \text{ is regular} \\ \gamma_i & \text{if observation } i \text{ is an outlier to be trimmed,} \end{cases} \tag{3.9}$$

where $\beta \in \mathbb{R}^p$ is the unknown parameter vector to be estimated, $\epsilon \in \mathbb{R}^n$ is the noise vector and $\gamma \in \mathbb{R}^n$ an intervention vector. A way to model doubtful observations to be trimmed is to introduce a vector $\tau \in \mathbb{R}^n$ modeling outliers:

$$\forall i \in \{1,\dots,n\}, \qquad \tau_i = \begin{cases} 0 & \text{if observation } i \text{ has to be taken into account} \\ y_i - x_i^t\beta - \epsilon_i & \text{if observation } i \text{ is an outlier to be trimmed,} \end{cases}$$

The model (3.9) can be rewritten as the following linear model [SO11]:

$$y = X\beta + \epsilon + \tau. \tag{3.10}$$

We are interested in minimizing the norm of the noise vector while selecting k_v variables and removing k_o outliers, that is, solving the following optimization problem [CCM13], for some $q \in \{1,2\}$,

$$\begin{cases} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n} & \frac{1}{q}\|X\beta+\tau-y\|_q^q \\ \text{s.t.} & \|\beta\|_0 \le k_v \\ & \|\tau\|_0 \le k_o. \end{cases} \tag{3.11}$$

This formulation allows the selection of relevant variables and the avoidance of outliers. When $k_o = 0$, no outlier detection is performed and this problem boils down to the best subset selection problem [Mil02, BKM16, MT15]. When $k_v = p$, no variable selection is performed, the resulting problem is known as the least trimmed squares regression problem [RL05, GP02]. Due to the nature of the cardinality constraints, Problem (4.4) is a non convex optimization problem and has been shown to be NP-hard and considered as an intractable problem. Mainstream research focused on solving a relaxed version of Problem (4.4), by using the ℓ_1 norm instead of the ℓ_0 norm:

$$\begin{cases} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n} & \frac{1}{2}\|X\beta+\tau-y\|_2^2 \\ \text{s.t.} & \|\beta\|_1 \le \lambda \\ & \|\tau\|_1 \le \gamma, \end{cases} \tag{3.12}$$

where λ and γ are two nonnegative regularization parameters. However, this approach is not globally optimal in the sense of (4.4). To retrieve the global minimum of Problem (4.4), we propose the use of mixed integer optimization (MIO). The MIO approach has a computational cost, but two decades of progress enabled its effective practical use for moderately sized problems. Furthermore, in a near future the use of high performance computing should allow some scalability [**?**, see for instance]for a more detailed motivation]bertsimas2016best.
We recall that the lagrangian relaxation of Problem (3.12) is given by:

$$\min_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n} \quad \frac{1}{2}\|X\beta+\tau-y\|_2^2 + \lambda||\beta||_1 + \gamma||\tau||_1. \tag{3.13}$$

Statistical properties of Problem (3.13) have been explored in ([DT19, NT12]).

3.6 Variable Selection and Outlier Detection as a MIO

We propose to reformulate Problem (4.4) as a mixed integer (binary) optimization problem (MIO) by introducing binary variables representing whether or not variables and observations are useful.

3.6.1 Introducing Binary Variables

Variable selection involves the ℓ_0 norm function to count the number of useful variables. This counting function can be represented by introducing p binary variables $z_j \in \{0,1\}$ such that

$$\|\beta\|_0 = \sum_{j=1}^{p} z_j \quad \text{and} \quad z_j = 0 \Leftrightarrow \beta_j = 0.$$

Different approaches can be used to force $z_j = 0 \Leftrightarrow \beta_j = 0$ into an optimization problem, such as:

1. Replace β_j by $z_j\beta_j$ for $j = 1,\dots,p$.
2. Set $|\beta_j|(1-z_j) = 0$ for $j = 1,\dots,p$ or $\sum_{j=1}^{p} |\beta_j|(1-z_j) = 0$.
3. Use a big-M constraint, $|\beta_j| \leq M_v z_j$ for $j = 1,\dots,p$ and for some fixed constant M_v large enough (such as $M_v \geq \max_j |\beta_j^\star|$, $\beta_j^\star$ being the solution of the optimization problem).
4. Treat $z_j = 0 \Leftrightarrow \beta_j = 0$ as logical implications (also called indicator constraints or special ordered set SOS-1). Note that this kind of logical implication can be efficiently handled in a branch-and-bound procedure for MIO problems.

We now discuss and give a short overview of the advantages and drawbacks of each approach. The two first approaches involve nonlinear interaction terms between binary and continuous variables. Their interest lies in the possibility of obtaining interesting continuous relaxations. The main advantage of the big M method (approach 3) is that it brings only linear inequality constraints, but the value of the M term needs to be chosen carefully since it shows a great deal of practical influence on the solver performance. Logical implications (approach 4) have the advantage of avoiding these types of problems, as they do not rely on a separate constant value. However, they tend to have weaker relaxations, a condition which may lead to longer solve times in a model. In this paper we will use the third approach for our implementation. Note that to be efficient, it is preferable to reformulate Problem (4.4) as a quadratic mixed binary program.

Outlier detection also involves the ℓ_0 norm function to count the number of outliers. This counting function can be represented by introducing n binary variables $t_i \in \{0,1\}$ such as

$$\|\tau\|_0 = \sum_{i=1}^{n} t_i \quad \text{and} \quad t_i = 0 \Leftrightarrow \tau_i = 0,\ (x_i, y_i) \text{ is not an outlier.}$$

3.6.2 A MIO Formulation

Introducing binary variables for both variables and outliers with two big M constraints, given appropriate parameters k_v, k_o, M_v and M_o, Problem (4.4) becomes for some $q \in \{1,2\}$:

$$\begin{aligned} \min_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n,z\in\{0,1\}^p,t\in\{0,1\}^n} \quad & \tfrac{1}{q}\|X\beta+\tau-y\|_q^q & & \\ \text{s.t.} \quad & \sum_{j=1}^{p} z_j \leq k_v \quad \text{and} \quad |\beta_j| \leq z_j M_v, & & j = 1,\dots,p \\ & \sum_{i=1}^{n} t_i \leq k_o \quad \text{and} \quad |\tau_i| \leq t_i M_o & & i = 1,\dots,n. \end{aligned} \tag{3.14}$$

This problem turns out to be a mixed binary quadratic program when $q = 2$. When $q = 1$, it can be written as a mixed binary linear program at the price of introducing $2n$ continuous positive variables as follows:

$$\begin{array}{lll} \min\limits_{\beta\in\mathbb{R}^p,\tau,\varepsilon^+,\varepsilon^-\in\mathbb{R}^n,z\in\{0,1\}^p,t\in\{0,1\}^n} & \sum\limits_{i=1}^{n}\varepsilon_i^+ + \varepsilon_i^- & \\ \text{s.t.} & \varepsilon_i^+ - \varepsilon_i^- = x_i^t\beta + \tau_i - y_i & i = 1,\dots,n \\ & \sum\limits_{j=1}^{p} z_j \le k_v & \\ & |\beta_j| \le z_j M_v & j = 1,\dots,p \\ & \sum\limits_{i=1}^{n} t_i \le k_o & \\ & |\tau_i| \le t_i M_o & i = 1,\dots,n \\ & 0 \le \varepsilon_i^+,\ 0 \le \varepsilon_i^- & i = 1,\dots,n. \end{array} \tag{3.15}$$

Note that the big M constraints can be replaced by special ordered set (SOS) constraints of type 1 leading to the following equivalent formulation with no more big M:

$$\begin{array}{lll} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n,z\in\{0,1\}^p,t\in\{0,1\}^n} & \frac{1}{q}\|X\beta + \tau - y\|_q^q & \\ \text{s.t.} & \sum\limits_{j=1}^{p} z_j \le k_v & \\ & (\beta_j, 1 - z_j) : \text{SOS} & j = 1,\dots,p \\ & \sum\limits_{i=1}^{n} t_i \le k_o & \\ & (\tau_i, 1 - t_i) : \text{SOS} & i = 1,\dots,n. \end{array} \tag{3.16}$$

It turns out that variable τ is not necessary and can be removed at the cost of the reintroduction of a big M constant as follows:

$$\begin{array}{lll} \min\limits_{\beta\in\mathbb{R}^p,\,\epsilon\in\mathbb{R}^n,\,z\in\{0,1\}^p,\,t\in\{0,1\}^n} & \frac{1}{q}\sum\limits_{i=1}^{n}|\varepsilon_i|^q & \\ \text{s.t.} & |x_i^t\beta + \epsilon_i - y_i| \le t_i M_o & i = 1,\dots,n \\ & \sum\limits_{j=1}^{p} z_j \le k_v & \\ & (\beta_j, 1 - z_j) : \text{SOS} & j = 1,\dots,p \\ & \sum\limits_{i=1}^{n} t_i \le k_o, & \end{array} \tag{3.17}$$

leading to, for $q = 2$, a MIQP with $p + n$ continuous variables, $p + n$ binary ones, $2n + 2$ inequality constraints and p SOS constraints.

3.6.3 Additional Constraints

One of the most important properties of a MIO formulation relies on the strength of its continuous relaxation. It is known that the knowledge of strong valid inequalities for the feasible sets may help the optimization process by providing tight lower bounds on the cost [WN88]. Such inequality can be provided by considering the convex hull of the feasible set

$$S = \Big\{\beta, \tau \;\Big|\; (z,t) \in \{0,1\}^{n+p},\ \sum_{j=1}^{p} z_j \le k_v,\ |\beta_j| \le z_j M_v,\ \sum_{i=1}^{n} t_i \le k_o,\ |\tau_i| \le t_i M_o \Big\},$$

that is

$$Conv(S) = \Big\{\beta, \tau \;\Big|\; ||\beta||_\infty \le M_v\ ||\tau||_\infty \le M_o\ ||\beta||_1 \le k_v M_v \text{ and } ||\tau||_1 \le k_o M_o \Big\}.$$

Adding the bounds of $||\beta||_\infty$, $||\tau||_\infty$, $||\beta||_1$ and $||\beta||_1$ typically leads to improved performance of the MIO, especially in delivering lower bound certificates [Vie15]. This remark leads us to consider for practical reasons the following completed mixed integer program with problem-dependent constants M_v and M_o. The choice of these constants is practically important ([BKM16]).

$$\begin{array}{lll}
\min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n,z\in\{0,1\}^p,t\in\{0,1\}^n} & \frac{1}{q}\|X\beta+\tau-y\|_q^q & \\
\text{s.t.} & \sum\limits_{j=1}^{p} z_j \le k_v, \qquad (\beta_j, 1-z_j) : \text{SOS} & j = 1,\dots,p \\
& \sum\limits_{i=1}^{n} t_i \le k_o,\ (\tau_i, 1-t_i) : \text{SOS} & i = 1,\dots,n \\
& \|\beta\|_1 \le k_v M_v,\ \ \|\beta\|_\infty \le M_v & \\
& \|\tau\|_1 \le k_o M_o,\ \ \|\tau\|_\infty \le M_o, &
\end{array} \tag{3.18}$$

Following [BKM16], we propose to specify those parameters from warm-start using (β_0, τ_0) the solution provided by a first order alternate projected gradient algorithm presented next section. Given (β_0, τ_0), a natural setting for these parameters is $M_v = \theta\|\beta_0\|$ and $M_o = \theta\|\tau_0\|$, for some $\theta \in [1,2]$.

Note that when $p > n$, as claimed in [BKM16], it may be advisable to also bound $X\beta$ by adding constraint of the type $\|X\beta\|_1 \le M_1$, $\|X\beta\|_\infty \le M_\infty$ in the formulation of the optimization problem. This brings two more constants M_1 and M_∞ to be initialized using the warm start (β_0, τ_0).

In the rest of the paper, the Formulation (4.7) with $q = 2$ is used to solve Problem (4.4). It will be denoted by ℓ_0 RR. Note that its ℓ_1 relaxation (Problem (3.12)) will be denoted by ℓ_1 RR.

3.7 Proximal Alternating Linearized Minimization Algorithm

In this section, an efficient alternate projected gradient algorithm providing a local solution to the optimization Problem (4.4) is introduced. Before entering into the details of the alternate projected gradient algorithm, it is appropriate to introduce the problem of finding the projection of a vector $u \in \mathbb{R}^p$ onto the set of $k \le p$ sparse vectors

$$\begin{array}{ll}
\min\limits_{v\in\mathbb{R}^p} & \frac{1}{2}\|v-u\|^2 \\
\text{s.t.} & \|v\|_0 \le k.
\end{array} \tag{3.19}$$

This problem is easy and its solution $v^\star$ is given by sorting on the absolute value of vector $|u|$, that is by a sequence of indices (j) such that $|u_{(1)}| \ge |u_{(2)}| \ge \dots |u_{(j)}| \ge \dots \ge |u_{(p)}|$.

Using these indices, the projection $v^\star = \mathrm{P}_k(u)$ of u is the vector u itself with its smallest coefficients set to 0 that is

$$v^\star = \mathrm{P}_k(u) = \begin{cases} u_j & \text{if } j \in \{(1), \dots, (k)\} \\ 0 & \text{else.} \end{cases}$$

We propose to use this projection mechanism, on both β and τ, to get a local solution to the initial Problem (4.4) at a low computing cost.

A possible way to achieve this goal consists of using the so-called block Gauss-Seidel iteration scheme on variables β and τ, also known as alternating minimization. To this end, a sequence $\{(\beta^\ell, \tau^\ell)\}_{\ell \in \mathbb{N}}$ is generated starting from some (β^0, τ^0) using the following scheme:

$$\begin{cases} \beta^{\ell+1} = arg\min\limits_{\beta \in \mathbb{R}^p} (\beta - \beta^\ell)^t \mathrm{X}^t (\mathrm{X}\beta^\ell + \tau^\ell - y) \\ \qquad\qquad \text{s.t.} \quad \|\beta\|_0 \le k_v \\ \qquad\qquad\quad \|\beta - \beta^\ell\|^2 \le d_v \end{cases} \quad \begin{cases} \tau^{\ell+1} = arg\min\limits_{\tau \in \mathbb{R}^n} (\tau - \tau^\ell)^t (\mathrm{X}\beta^{\ell+1} + \tau^\ell - y) \\ \qquad\qquad \text{s.t.} \quad \|\tau\|_0 \le k_o \\ \qquad\qquad\quad \|\tau - \tau^\ell\|^2 \le d_o. \end{cases}$$

Where d_v and d_o are two given positive parameters that can be changed each step. The idea of the proximal method is, at each iteration, to minimize a regularized first-order approximation of the cost that can be interpreted as a local trust region mechanism [**?**, for details see for instance]]parikh2014proximal. This surrogate loss is also a local upper bound of the targeted loss since, for well chosen ρ_v and ρ_o, the Lagrange multipliers associated with the trust region constraints are

$$\begin{cases} \frac{1}{2}\|\mathrm{X}\beta + \tau^\ell - y\|^2 \le \frac{1}{2}\|\mathrm{X}\beta^\ell + \tau^\ell - y\|^2 + (\beta - \beta^\ell)^t \mathrm{X}^t (\mathrm{X}\beta^\ell + \tau^\ell - y) + \dfrac{1}{2\rho_v}\|\beta - \beta^\ell\|^2 \\ \frac{1}{2}\|\mathrm{X}\beta^{\ell+1} + \tau - y\|^2 \le \frac{1}{2}\|\mathrm{X}\beta^{\ell+1} + \tau^\ell - y\|^2 + (\tau - \tau^\ell)^t (\mathrm{X}\beta^{\ell+1} + \tau^\ell - y) + \dfrac{1}{2\rho_o}\|\tau - \tau^\ell\|^2. \end{cases}$$

For each iteration, this method introduced by [BST14] and called the proximal alternating linearized minimization (PALM) algorithm, consists of minimizing the upper bounds as follows:

$$\begin{cases} \beta^{\ell+1} = arg \min\limits_{\beta \in \mathbb{R}^p, \|\beta\|_0 \le k_v} (\beta - \beta^\ell)^t \mathrm{X}^t (\mathrm{X}\beta^\ell + \tau^\ell - y) + \dfrac{1}{2\rho_v}\|\beta - \beta^\ell\|^2 \\ \tau^{\ell+1} = arg \min\limits_{\tau \in \mathbb{R}^n, \|\tau\|_0 \le k_o} (\tau - \tau^\ell)^t (\mathrm{X}\beta^{\ell+1} + \tau^\ell - y) + \dfrac{1}{2\rho_o}\|\tau - \tau^\ell\|^2. \end{cases}$$

That is, after some algebra,

$$\begin{cases} \beta^{\ell+1} = arg \min\limits_{\beta \in \mathbb{R}^p, \|\beta\|_0 \le k_v} \frac{1}{2}\|\beta - \beta^\ell + \rho_v \mathrm{X}^t (\mathrm{X}\beta^\ell + \tau^\ell - y)\|^2 \\ \tau^{\ell+1} = arg \min\limits_{\tau \in \mathbb{R}^n, \|\tau\|_0 \le k_o} \frac{1}{2}\|\tau - \tau^\ell + \rho_o (\mathrm{X}\beta^{\ell+1} + \tau^\ell - y)\|^2. \end{cases}$$

These two minimization problems are of the same kind as Problem (4.8) and thus the sequence can be generated by using two ℓ_0 projected gradient, that is:

$$\begin{cases} \beta^{\ell+1} = \mathrm{P}_{k_v}\big(\beta^\ell - \rho_v \mathrm{X}^t (\mathrm{X}\beta^\ell + \tau^\ell - y)\big) \\ \tau^{\ell+1} = \mathrm{P}_{k_o}\big(\tau^\ell - \rho_o (\mathrm{X}\beta^{\ell+1} + \tau^\ell - y)\big). \end{cases}$$

Algorithm 4.1 presents the pseudo code of the PALM algorithm.

Algorithme 3.2 : Proximal alternating linearized minimization (PALM) [BST14].

Data : X, y initialization $\beta, \tau = 0$
Result : β, τ
set $\rho_v \leq \frac{1}{\sigma_M^2}$ and $\rho_o \leq 1$
while *it has not converged* $(||\beta_{n+1} - \beta_n||_2 > 10^{-6})$ **do**
  $d \leftarrow \beta - \rho_v X^\top (X\beta + \tau - y)$ — variable selection
  $\beta \leftarrow P_{k_v}(d)$
  $\delta \leftarrow \tau - \rho_o (X\beta + \tau - y)$ — eliminating outliers
  $\tau \leftarrow P_{k_o}(\delta)$

This algorithm converges towards a local minima of Problem (4.4) since it fulfills the assumption needed for Theorem 3.1 in [BST14]. Indeed, the partial gradients $G_\beta(\beta) = X^\top(X\beta + \tau - y)$ and $G_\tau(\tau) = (X\beta + \tau - y)$ are globally Lipschitz with module respectively $\frac{1}{\sigma_M^2}$ and 1, σ_M being the largest singular value of X. Also, to prove the convergence, the stepsizes have to be chosen such that $\rho_v \leq \frac{1}{\sigma_M^2}$ and $\rho_o \leq 1$.

3.8 Results for Synthetic Data Sets

In this section we show the empirical performance of the MIO approach.

3.8.1 Setup

In [HTT], a follow-up paper to ([BKM16]), the authors provide a synthetic setup considering a wide range of SNR values. We use it here to compare the best subset selection (Formulation (4.7) with $k_o = 0$), the lasso, PALM, the ℓ_0 robust regression - ℓ_0 RR and the ℓ_1 robust regression - ℓ_1 RR. The same notations as [HTT] were used, namely n, p (problem dimensions), s (sparsity level), beta-type (pattern of sparsity), ρ (predictor autocorrelation level), and ν (SNR level).

— We define coefficients $\beta_0 \in \mathbb{R}^p$ according to s and the beta-type, as described below.

— We draw the rows of the matrix $X \in \mathbb{R}^{n\times p}$ from $N_p(0, \Sigma)$, where $\Sigma \in \mathbb{R}^{p\times p}$ has entry (i, j) equal to $\rho^{|i-j|}$, and $\rho = 0.35$.

— We draw the vector $y \in \mathbb{R}^n$ from $N_n(X\beta_0, \sigma^2 I)$, with σ^2 defined to meet the desired SNR level, i.e., $\sigma^2 = \beta_0^T \Sigma \beta_0 / \nu$.

— We use 5-fold cross validation and the tuning was performed by minimizing prediction error on the test set.

— To assess the influence of outliers, 5% of outliers were added to the data set by following a normal $N(50, \sigma)$ instead of $N(0, \sigma)$.

— We considered two configurations: the low setting with $n = 150$, $p = 15$, and the medium setting $n = 500$, $p = 100$. For each configuration, we also considered two settings: the first one with outliers generated as mentioned above, and the second one without adding outliers.

— The lasso was tuned over 100 values of λ (as it is in `glmnet`).

— In order to determine the values of k_v, M_v, k_o and M_o, we run the PALM algorithm for k_v ranging from 1 to p and for k_o ranging from 0 to 10% with a step size of 2.5%. Then, we choose the solution with the minimal error $||X_{test}\beta_{palm} - y_{test}||_2^2$.

— $M_v = (1+\alpha)||\beta_{palm}||_\infty$, $M_o = (1+\alpha)||\tau_{palm}||_\infty$ with $\alpha = 0.1$, k_v and k_o are set as the number of nonzero elements in the solutions β_{palm} and τ_{palm} respectively.

— The ℓ_1 robust regression (ℓ_1 RR) algorithm was tuned over five values of λ from zero to $1.5||\beta_{lasso}||_\infty$ where β_{lasso} is the solution obtained by the lasso method, and over fifty one values of γ from 0 to 5000 with a step size of 100 for the low dimensional case, and from 0 to 10000 with a step size of 200 for the medium dimensional case.

— We run the best subset selection, the lasso, PALM, the ℓ_0 robust regression (ℓ_0 RR) the ℓ_1 robust regression (ℓ_1 RR) using a 5-fold cross validation. The tuning was performed by minimizing the error on the test set.

— We repeat 10 times for the low dimensional setting and 5 times for the medium dimensional setting and average the results.

Coefficients: We considered three settings for the coefficients $\beta_0 \in \mathbb{R}^p$ as in [HTT]:

— beta-type 1: $\beta_0 = (1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ \underbrace{0,\ldots,0}_{p-10 \text{ times}})$;

— beta-type 2: β_0 has its first 5 components equal to 1, and the rest equal to 0;

— beta-type 5: β_0 has its first 5 components equal to 1, and the rest decaying exponentially to 0, specifically, $\beta_{0i} = 0.5^{i-s}$, for $i = s+1,\ldots,p$, where $s = 5$;

Following [BKM16, HTT], we use, as an accuracy metric, the relative risk (RR) defined by:

$$\mathrm{RR}(\hat{\beta}) = \frac{\mathbb{E}(x_0^T\hat{\beta} - x_0^T\beta_0)^2}{\mathbb{E}(x_0^T\beta_0)^2} = \frac{(\hat{\beta}-\beta_0)^T\Sigma(\hat{\beta}-\beta_0)}{\beta_0^T\Sigma\beta_0},$$

The best score is 0 (when $\hat{\beta} = \beta_0$) and the null score is 1, obtained when $\hat{\beta} = 0$.
We also use the proportion of variance explained (PVE) defined by:

$$\mathrm{PVE}(\hat{\beta}) = 1 - \frac{\mathbb{E}(y_0 - x_0^T\hat{\beta})^2}{\mathrm{Var}(y_0)} = 1 - \frac{(\hat{\beta}-\beta_0)^T\Sigma(\hat{\beta}-\beta_0)+\sigma^2}{\beta_0^T\Sigma\beta_0+\sigma^2}.$$

The maximum value for the PVE, also called the perfect score, is SNR/(SNR+1) (see [HTT] for details).

3.8.2 Computational Costs

For the lasso, we used the **Matlab** "lasso" function with 100 values of λ as implemented in **glmnet**. The solution is delivered in a very short time. For the best subset selection problem, we implemented the method using the MIO Formulation (4.7) with $k_o = 0$, used PALM to compute a warm start and then call Gurobi through its Matlab interface. We used a time limit of 3 minutes for Gurobi to optimize the best subset selection problem for both low and medium dimensional case. The same procedure is followed for the ℓ_0 robust regression problem but with a time limit increased to 10 minutes for the medium dimensional setting.
For the ℓ_1 robust regression, we obtained $5 \times 51 = 255$ (5 values of λ and 51 values of γ) solutions for each test. The time needed to obtain each solution depends on the size of the dataset, but it varies from 0.16 second to about 1 second.
We can conclude that for low dimensional setting, we faced around 15 hours of computation , and more than 45 hours for the medium dimensional setting for each type of β.

3.8.3 Results

Figures (3.1)-(3.6) plot the relative risk (left panel) and the proportion of variance explained (right panel) as functions of signal-to-noise ratio (SNR). The results can be divided into two main categories:

No Outliers:

In this case, no outliers were added to the synthetic data sets generated as mentioned before. Figures (3.1), (3.2), (3.3), (3.4), (3.5) and (3.6) show that for small SNR values, the ℓ_1 methods (lasso and ℓ_1 RR) have the lead on the other methods (best subset selection, PALM and ℓ_0 RR). While for high SNR values the ℓ_0 approaches outperform the ℓ_1 approaches even though all the methods perform quite similarly for high SNR values.

Presence of Outliers

In this case, Figures (3.1), (3.2), (3.3), (3.4), (3.5) and (3.6) show that PALM, ℓ_0 RR and ℓ_1 RR outperform the best subset selection and the lasso, which is not surprising since the last two methods are not robust to outliers. In addition, for SNR < 0.25 the ℓ_1 RR performs, in general, better than PALM and the ℓ_0 RR. But for higher SNR values, there is no clear winner. An important caveat to emphasize up front is that the Gurobi MIO algorithm for ℓ_0 RR was given only 10 minutes per problem, which may have caused the ℓ_0 RR to underperform, and that the performance of the MIO algorithm depends on the parameters tuned using PALM.

3.8.4 Detection Rate for the Reature Selection and Outlier Detection Tasks

To determine whether the ℓ_0 robust regression approach can detect the outliers and select the right features, we generated two low dimensional and two medium dimensional data sets using the β type-2, with SNR values 0.5 and 5. We added 5% of outliers in the response vector (as in the setup of the synthetic data sets). k_v and k_o were set as the true sparsity level of β and as the percentage of outliers (5%). In all cases, the detection rate of both outliers and features was 100%, noting that no cross validation was performed.
In the experiments performed on both real and data sets, we used PALM to tune the parameters k_v and k_o. Thus the performance of the MIO approach depends on PALM. To this end, each data set was split into two parts: the training set (70%) and the testing set (30%). We added 5% of outliers in the training set's response vector. PALM was performed for $k_v \in [1,\dots,p]$ and $\frac{k_o}{n} \in [0, 0.025, 0.05, 0.075, 0.1]$. PALM failed to estimate the true sparsity level and the percentage of outliers, it overestimated the values of the parameters. This leads the PALM-MIO approach to fail at detecting the percentage of outliers and selecting the relevant features, even though all the true outliers were considered as outliers by this approach.

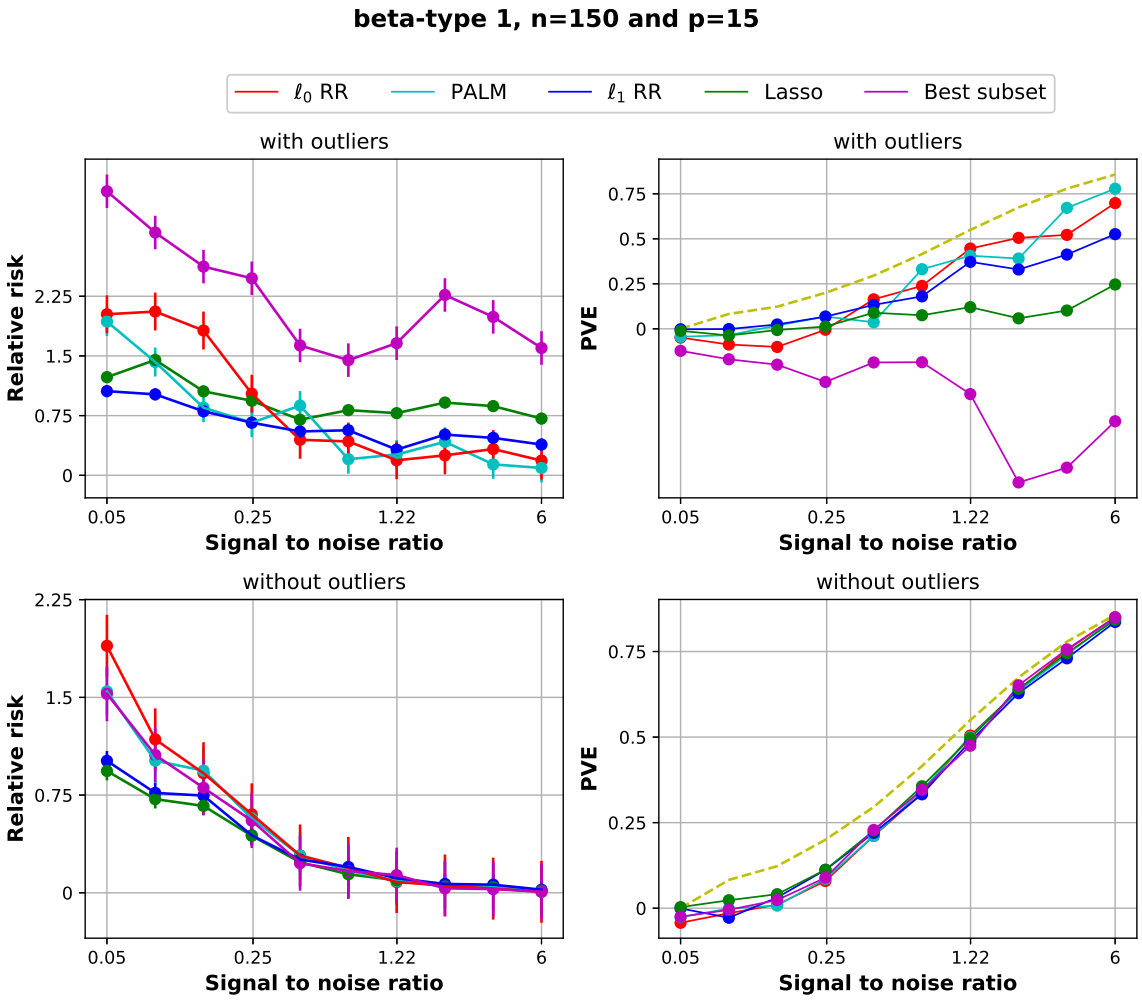

FIGURE 3.1 – Relative risk (left panel) and proportion of variance explained (right panel) functions of SNR, for beta-type 1 in the setting with n = 150, p = 15, and s = 5 with and without outliers (top panel and bottom panel respectively).

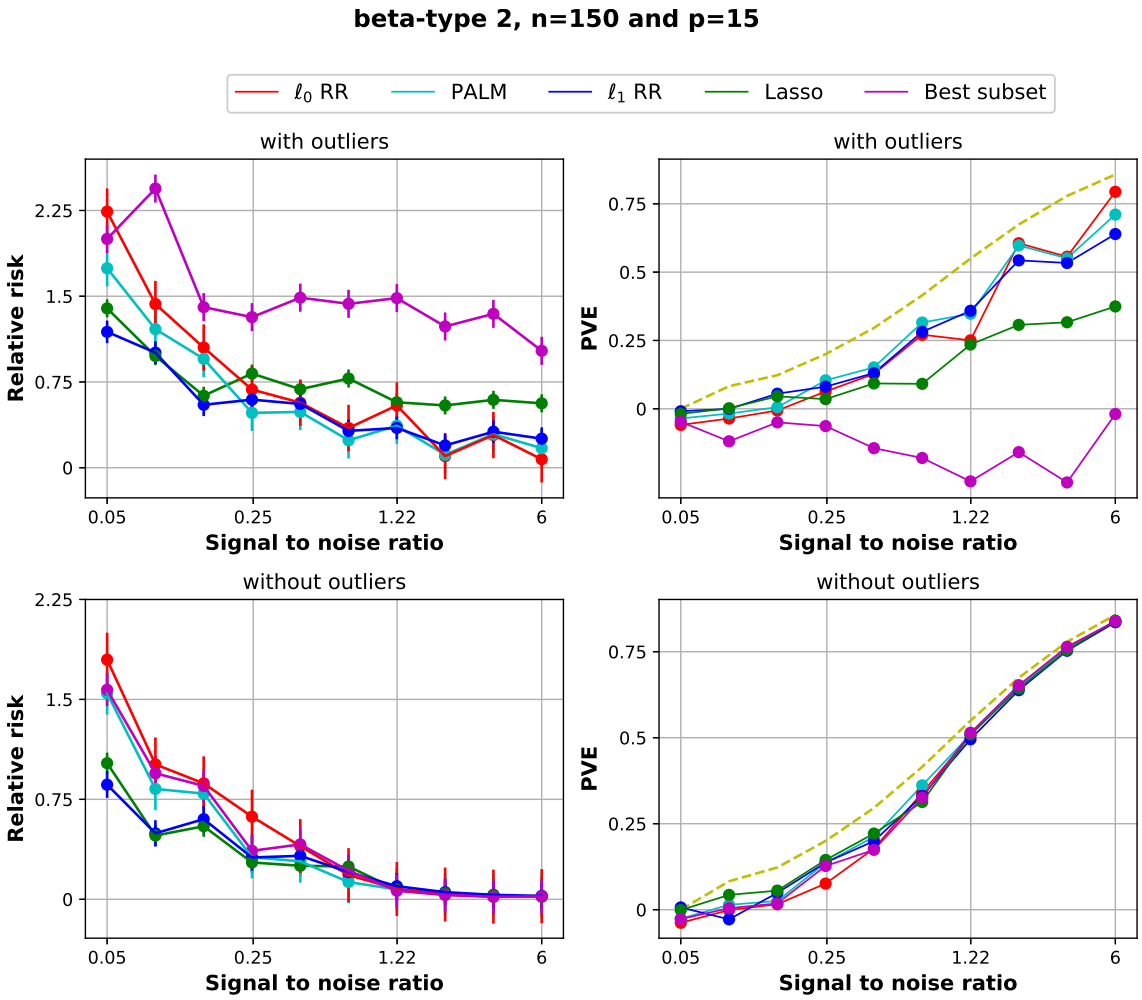

FIGURE 3.2 – Relative risk (left panel) and proportion of variance explained (right panel) functions of SNR, for beta-type 2 in the setting with n = 150, p = 15, and s = 5 with and without outliers (top panel and bottom panel respectively).

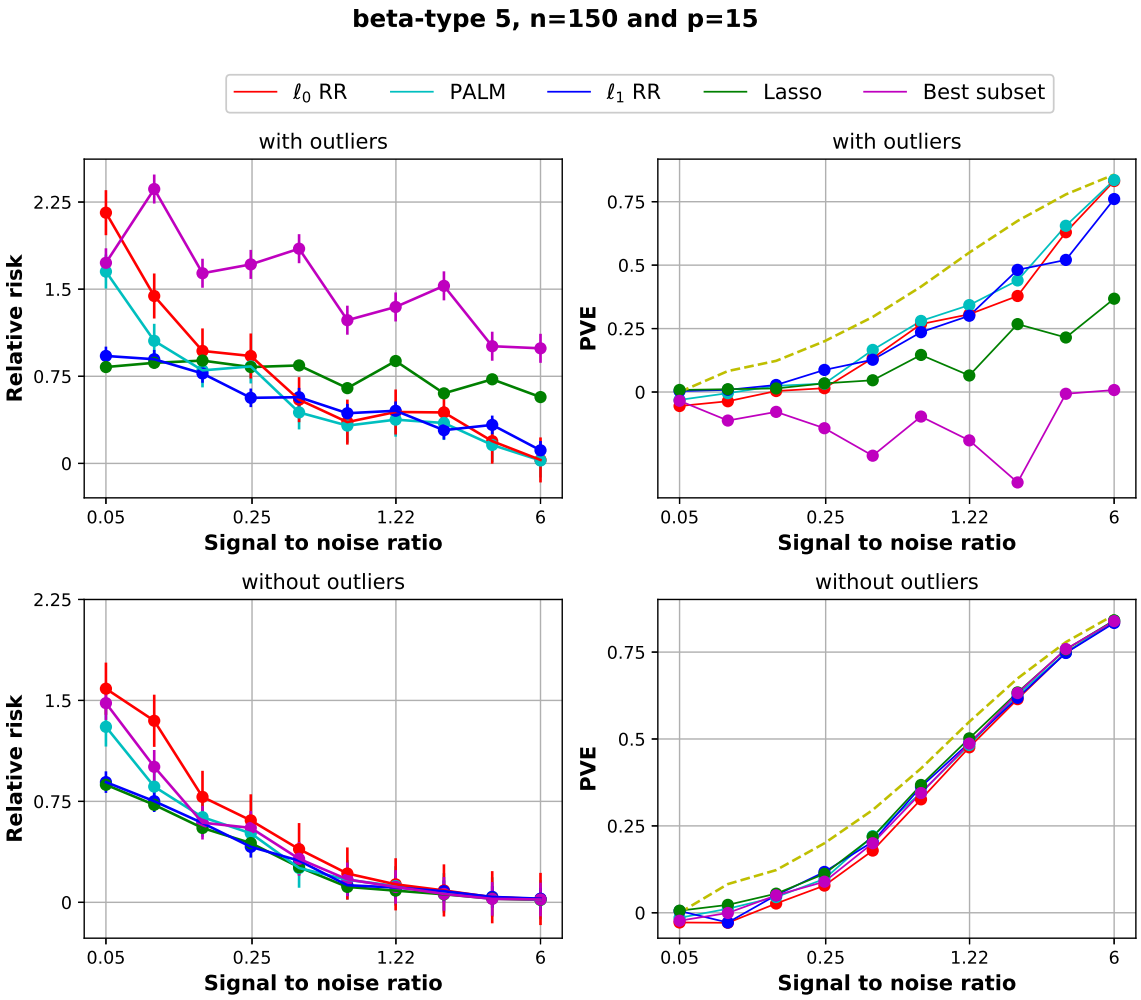

FIGURE 3.3 – Relative risk (left panel) and proportion of variance explained (right panel) functions of SNR, for beta-type 5 in the setting with n = 150, p = 15, and s = 5 with and without outliers (top panel and bottom panel respectively).

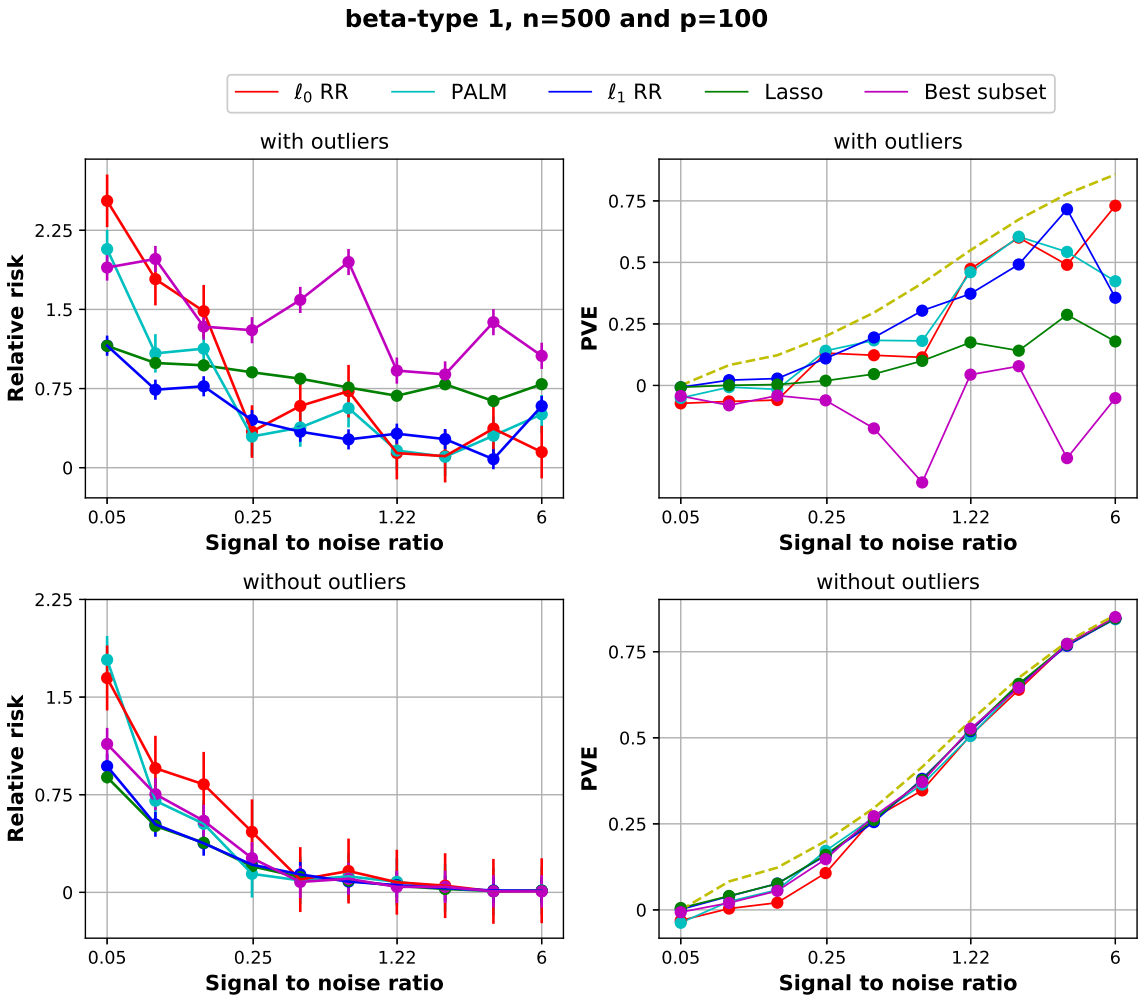

FIGURE 3.4 – Relative risk (left panel) and proportion of variance explained (right panel) functions of SNR, for beta-type 1 in the setting with n = 500, p = 100, and s = 5 with and without outliers (top panel and bottom panel respectively).

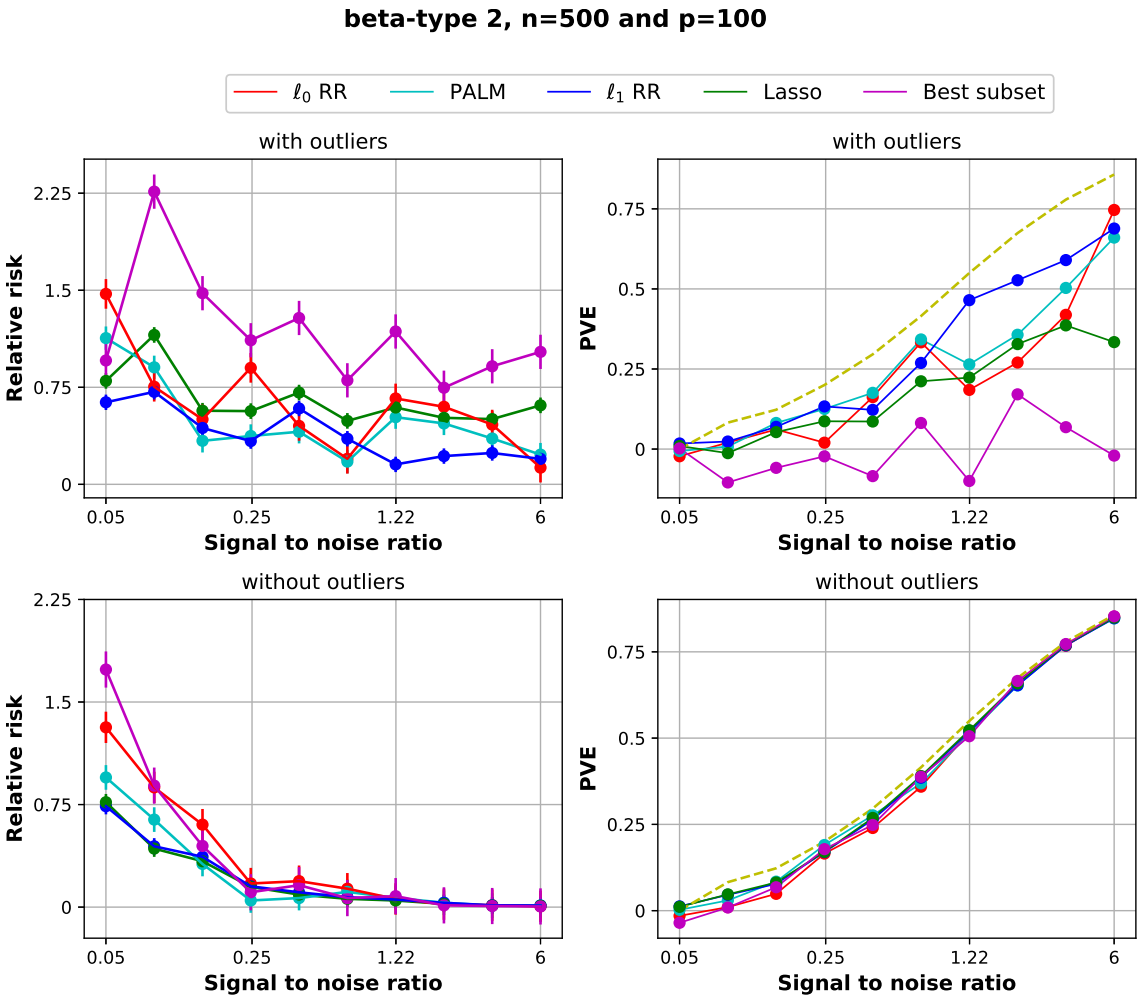

FIGURE 3.5 – Relative risk (left panel) and proportion of variance explained (right panel) functions of SNR, for beta-type 2 in the setting with n = 500, p = 100, and s = 5 with and without outliers (top panel and bottom panel respectively).

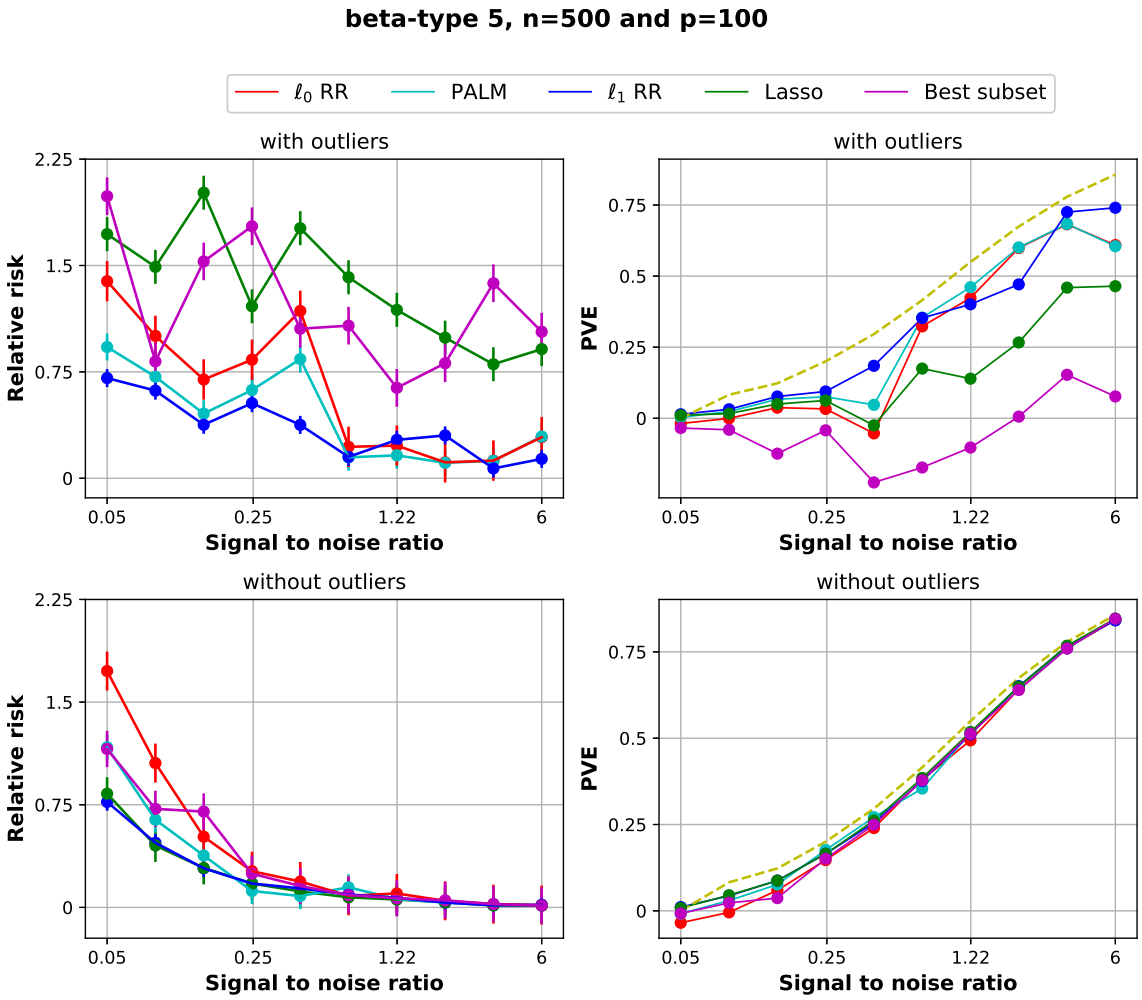

FIGURE 3.6 – Relative risk (left panel) and proportion of variance explained (right panel) functions of SNR, for beta-type 5 in the setting with n = 500, p = 100, and s = 5 with and without outliers (top panel and bottom panel respectively).

TABLE 3.3 – Summary of used datasets.

Name of the dataset	*number of instances n*	*number of attributes p*	*Origin*
Body Fat	252	15	`lib.stat.cmu.edu`
Concrete Compressive Strength	1030	9	UCI
Concrete Slump Test	103	10	UCI
Real Estate Valuation	414	7	UCI
Diabetes	442	10	`stat.ncsu.edu`
Boston Housing	489	3	Web [1]
Auto Mpg	398	8	UCI

3.9 Real Data Sets

The performances of all methods have been compared on real data sets. To this end we have used 7 data sets presented in Table 4.1. The different methods have been compared on all these data sets according to the following setup:

— The response vector y and the columns of the matrix X have been normalized and standardized to have zero mean and one standard deviation;

— Two 5-fold cross validation loops have been implemented. The inner one has been used to give a relevant choice for the hyper-parameters. The outer one has been used to estimate the average mean squared error MSE;

— As for synthetic data sets, we run PALM for k_v ranging from 1 to p, and k_o ranging from 0 to 10% with a step seize of 2.5%, and pick the solution with smallest cross validation error. This obtained solution is used to set the values of M_v and M_o and as a warm start for the ℓ_0 robust regression algorithm as well;

— The hyper-parameter λ of the lasso was tuned over 100 values as per the default in `glmnet`;

— The ℓ_1 robust regression algorithm was tuned over 5 values of λ (as for the synthetic data sets) and over 40 values of γ varying from 0 to 2000 with a step size of 50. We remarked that, for the normalized and standardized data set considered, it's enough to bound $||\tau||_1$ by 2000;

— Outliers were generated by replacing 5% of the response vector values y_i by $y_i + 2(\max(y) - \min(y))$ that is a constant value set to the range of the response variable in the training set;

Each experience is repeated 3 times. Tables 3.4 and 3.5 report the average of the results and the standard deviation in parentheses for the raw data.

An important caveat to emphasize upfront is that the ℓ_0 robust regression algorithm was given 10 minutes time limit per problem instance per subset size. This practical restriction may have caused this algorithm to under perform in some cases. For the best subset selection problem, the time limit was set to 2 minutes. We note that the optimality was certified for almost every case in less than two minutes. In the absence of outliers, results in Table 3.4 show that there is no clear winner. It is remarkable that all methods performed quite similarly, with a little advantage of using the lasso. In the presence of outliers, results in Table 3.5 show the dominance of the robust regression algorithms used over the best subset selection and the lasso. The ℓ_0 robust regression performed better than the other methods.

TABLE 3.4 – Cross Validation MSE Rates (Standard Deviations) of the Best subset, Lasso, PALM, ℓ_0 Robust Regression (ℓ_0 RR) and ℓ_1 Robust Regression (ℓ_1 RR) on 7 Real Datasets.

	Best subset	Lasso	Palm	ℓ_0 RR	ℓ_1 RR
Body Fat	**2.2797** ($7.2e^{-5}$)	4.2644 ($1.5e^{-4}$)	2.5958 ($5.2e^{-5}$)	2.6270 ($4.77e^{-5}$)	4.5008 ($6.2e^{-5}$)
Concrete Compressive Strength	**0.3588** (0.018)	0.3602 (0.019)	0.3692 ($4.2e^{-4}$)	0.3693 ($3.5e^{-4}$)	0.3603 (0.015)
Slump Test	0.0880 (0.008)	**0.0863** (0.012)	0.0864 (0.011)	0.0880 (0.008)	0.0869 (0.010)
Real Estate Valuation	0.2994 (0.024)	**0.2924** (0.036)	0.3010 (0.026)	0.2992 (0.026)	0.2950 (0.033)
Diabetes	0.3917 (0.037)	0.3914 (0.038)	0.3889 (0.028)	**0.3888**(0.038)	0.3952 (0.039)
Boston Housing	0.2460 (0.007)	0.2460 (0.007)	0.2446 (0.008)	**0.2440** (0.009)	0.2448 (0.006)
Auto Mpg	0.1469 (0.002)	**0.1458** (0.005)	0.1523 (0.007)	0.1516 (0.007)	0.1478 (0.008)

TABLE 3.5 – Cross validation MSE rates (standard deviations) of the of the Best subset, Lasso, PALM, ℓ_0 Robust Regression (ℓ_0 RR) and ℓ_1 Robust Regression (ℓ_1 RR) on 7 Real Datasets Corrupted by 5% of Outliers in the Initial Response Vector y.

	Best subset	Lasso	Palm	ℓ_0 RR	ℓ_1 RR
Body Fat	0.3923 (0.023)	0.4039 (0.034)	**0.3679** (0.024)	0.3764 (0.009)	0.3882 (0.023)
Concrete compressive strength	0.5891 (0.063)	0.5877 (0.059)	0.5843 (0.070)	**0.5842** (0.071)	0.5857 (0.755)
Slump test	0.2749 (0.186)	0.2463 (0.128)	0.1110 (0.022)	**0.0958** (0.012)	0.1039 (0.018)
Real estate valuation	0.6581 (0.131)	0.6680 (0.146)	0.6587 (0.137)	**0.6580** (0.138)	0.6688 (0.147)
Diabetes	0.5087 (0.015)	0.5002 (0.011)	0.5012 (0.009)	0.5009 (0.011)	**0.4923** (0.014)
Boston housing	0.5408 (0.240)	0.5293 (0.231)	0.5425 (0.241)	0.5441 (0.241)	**0.5235** (0.225)
Auto mpg	0.5498 (0.139)	0.5596 (0.128)	0.5406 (0.160)	0.5406 (0.160)	**0.5370** (0.163)

3.10 Conclusion

In this chapter, we propose a method for linear regression which solves the underlying optimization problem that handles both variable selection and outlier detection. We formulate the problem as a mixed-integer optimization problem and present a fast alternating minimization algorithm to find local minima. Furthermore, we present an empirical comparison between this method and its ℓ_1 relaxation on both synthetic and real data. We have found that neither the ℓ_0 norm problem nor its ℓ_1 relaxation dominates the other. Our recommendation is to use the ℓ_0 norm problem for large SNR while ℓ_1 relaxation is preferred when SNR is small. While the ℓ_0 approach is considered to be intractable, especially, for high dimensional regimes, one can propose to use screening rules helping in accelerating the solvers. Moreover, we have shown that if the true number of features and percentage of outliers are well esitmated, the speed of convergence to the global minimum decreases significantly. Furthermore, for high dimensional data sets, if we desire to obtain near optimal solutions is a short time, we suggest the use of the first order discrete optimization algorithm (PALM) which has shown high efficiency in terms of prediction error and computational cost.

Chapitre 4

ℓ_1 Regularized Robust and Sparse Linear Regression

Sommaire

4.1 Introduction

In regression, feature selection is an effective strategy to handle contaminated data and to deal with high dimensionality while providing better prediction. In addition to the presence of spurious variables, estimators suffer form corrupted, incorrectly measured or misreported observations known as outliers. The natural way to select relevant variables and to detect outliers is done by using the ℓ_0 norm for both aspects and recast the obtained optimization problem as a mixed integer optimization (MIO) problem. The ℓ_0 norm estimators perform well when the signal to noise ratio (SNR) is high. However, its performance decreases when the SNR is low due to the overfitting behavior of the ℓ_0 norm when the noise is relatively high. To fix this problem, we propose to regularize the ℓ_0 norm problem for variable selection and outlier detection by adding an ℓ_1 penalty term. We also propose an efficient and scalable non-convex proximal alternate algorithm producing high quality solution in a short time and used as a warm start for the MIO solver.

An empirical comparison between the ℓ_0 norm approach and its ℓ_1 regularized extension is presented as well. Results provided that the MIO regularized approach and its discrete first order warm start provide high quality solutions and performs better then the ℓ_0 approach especially for low SNR values. We consider the linear regression model:

$$y = X\beta + \epsilon\,,$$

where $y \in \mathbb{R}^n$ is the response vector, $X \in \mathbb{R}^{n\times p}$ is the model matrix, $\beta \in \mathbb{R}^p$ is the vector of regression coefficients and $\epsilon \in \mathbb{R}^n$ is the error vector. We assume that the columns of X have been standardized to have zero means and unit ℓ_2-norm.

In high-dimensional regimes i.e $p \gg n$, it is desired to estimate β by a sparse vector, that is a vector with few nonzero elements. To this end, feature selection has been of great importance in the last few decades [Mil02]. A natural way to compute sparse regression coefficients is to solve the, well known, best subset selection problem:

$$\begin{array}{ll}\min\limits_{\beta\in\mathbb{R}^p} & \frac{1}{2}\|X\beta - y\|_2^2 \\ \text{s.t.} & \|\beta\|_0 \le k_v.\end{array} \tag{4.1}$$

Where the ℓ_0 norm of a vector β counts the number of nonzeros in β. This classical problem dates back to at least [BKM67, HL67]. It has been considered as intractable since it is an NP-hard problem [Nat95, Mil02]. To overcome the computational difficulty of the best subset selection, [Tib96] proposed an ℓ_1 relaxation of the cardinality constraint, widely known as the "lasso":

$$\begin{array}{ll}\min\limits_{\beta\in\mathbb{R}^p} & \frac{1}{2}\|X\beta - y\|_2^2 \\ \text{s.t.} & \|\beta\|_1 \le k\,.\end{array} \tag{4.2}$$

The popularity of Lasso is due to its computational feasibility with the guarantee of getting a sparse model with good predictive performance. There have been an impressive amount of works studying statistical properties of Lasso and proposing algorithms to solve it [EHJ$^+$04, FHT10, FHST16, SBC$^+$17, EHJ$^+$04] and the books or surveys [BVDG11, FHT01, Tib11].

Throughout the years, researchers thought that best subset selection should be used whenever it is possible. Unfortunately, best subset selection is NP-hard and popular implementations such as the **R** package leaps do not scale to problem sizes larger than $p = 30$. To this end, Problem (4.1) was considered to be an intractable problem. In its work, [BKM15] showed that it is possible to find near optimal solution for high dimensional regimes in minutes (even though it takes hours to prove optimality) by formulating it as a mixed integer optimization problem (MIO). They also claimed that the best subset selection performs better than its competitors (lasso for example). However, this claim was refuted by [**?**], and best subset selection is no more the "holy grail" estimator for sparse modeling in regression: it overperforms other estimators for high signal to noise ratio (SNR) values, while lasso ensures better predictive performance for low SNR values. In fact, best subset selection suffers from overfitting. To overcome the overfitting of the best subset selection, [MRD17] suggested to add an ℓ_q penalty term to the objective function of Problem (4.1), where $q \in \{1,2\}$, so that the obtained problem is the following:

$$\begin{array}{ll}\min\limits_{\beta\in\mathbb{R}^p} & \frac{1}{2}\|X\beta - y\|_2^2 + \lambda||\beta||_q^q \\ \text{s.t.} & \|\beta\|_0 \le k_v.\end{array} \tag{4.3}$$

The proposed method *mitigates, to a large extent, the poor predictive performance of best-subsets in the low SNR regimes.*

The quality of estimators is known to be very sensitive to the presence of corrupted observations (outliers). Dealing with the presence of outliers can be divided into two categories: (a) the so-called "robust statistics", that is robust-to-outlier loss functions and (b) outlier detection per se which exclude outliers from the training set. In category (a), [RL05] is a relevant reference (see for instance chapters 3,6 and 7). However, in category (b), [CCM13, Li13, LDB09, SO11] are worth mentioning. For both categories (a) and (b) [RH18, HA04] offer comprehensive references.

To solve the robust sparse regression problem, [WLJ07] proposed the least absolute deviation (LAD) loss to deal with outliers, together with the ℓ_1 Lasso penalty, that is for $\lambda \geq 0$:

$$\min_{\beta} \|X\beta - y\|_1 + \lambda\|\beta\|_1.$$

Later, [LZ08, WWL12] showed that this problem was a particular case of a more general class of problems. To model outliers a sparse variable τ was introduced in [WYG$^+$08] for the first time. The idea was to minimize $||\beta||_1 + ||\tau||_1$ s.t $y = X\beta + \tau$. This formulation is the convex relaxation of minimizing $||\beta||_0 + ||\tau||_0$ s.t $y = X\beta + \tau$ as claimed by [WMM$^+$10]. The same idea was developped in many works [LDB09, NT12] for example. Later and for the first time, the use of the ℓ_0 norm for both variables β and τ was proposed by [CCM13]. They proposed the brute force algorithm defined, for some $q \in \{1,2\}$, by:

$$\begin{array}{ll} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n} & \frac{1}{q}\|X\beta + \tau - y\|_q^q \\ \text{s.t.} & \|\beta\|_0 \leq k_v \\ & \|\tau\|_0 \leq k_o, \end{array} \tag{4.4}$$

This formulation allows the selection of relevant variables and the avoidance of outliers, it can be solved by recasting it as a mixed integer optimization problem. Furthermore, as it is consists of double ℓ_0 norms, this formulation performs well for high SNR values, and its performance degrades as the SNR value decreases. To this end, we propose to regularize Problem (4.4) by adding an ℓ_1 penalty term as done in [MRD17] to overcome the limitation of the ℓ_0 norm when the SNR is low. The proposed problem is defined by:

$$\begin{array}{ll} \min\limits_{\beta\in\mathbb{R}^p,\ \tau\in\mathbb{R}^n} & \dfrac{1}{2}||y - X\beta - \tau||_2^2 + \lambda||\beta||_1 \\ \text{s.t} & ||\beta||_0 \leq k_v \\ & ||\tau||_0 \leq k_o. \end{array} \tag{4.5}$$

Contributions:

We summarize the contributions of this chapter below:

1. We propose to recast Problem (4.5) as a mixed integer optimization problem and use an efficient solver to solve it. Note that the suboptimality of the obtained solution is guaranteed even if we terminate the algorithm early,
2. We introduce an algorithm based on a discrete extension of first order continuous optimization methods. This framework is scalable and provides a local solution often close to the global one.
3. We propose to accelerate the MIO by using the discrete first order algorithm as a warm start.
4. We present computational results on both real and synthetic data sets. The results show that the proposed method performs well for low and high SNR values.

4.2 Mixed Integer Optimization Formulation

Binary variables will be introduced to reformulate Problems (4.4) and (4.5) as mixed integer binary optimization problems. These binary variables represent whether or not variables and observations are useful.

4.2.1 Introducing Binary Variables

Variable selection involves the ℓ_0 norm function to count the number of useful variables. This counting function can be represented by introducing p binary variables $z_j \in \{0,1\}$ such that

$$\|\beta\|_0 = \sum_{j=1}^{p} z_j \quad \text{and} \quad z_j = 0 \Leftrightarrow \beta_j = 0.$$

Different approaches can be used to force $z_j = 0 \Leftrightarrow \beta_j = 0$ into an optimization problem, such as:

1. Replace β_j by $z_j\beta_j$ for $j = 1,\dots,p$.
2. Set $|\beta_j|(1-z_j) = 0$ for $j = 1,\dots,p$ or $\sum_{j=1}^{p} |\beta_j|(1-z_j) = 0$.
3. Use a big-M constraint, $|\beta_j| \leq M_v z_j$ for $j = 1,\dots,p$ and for some fixed constant M_v large enough (such as $M_v \geq \max_j |\beta_j^\star|$, $\beta_j^\star$ being the solution of the optimization problem).
4. Treat $z_j = 0 \Leftrightarrow \beta_j = 0$ as logical implications (also called indicator constraints or special ordered set SOS-1). Note that this kind of logical implication can be efficiently handled in a branch-and-bound procedure for MIO problems.

We now discuss and give a short overview of the advantages and drawbacks of each approach. The two first approaches involve nonlinear interaction terms between binary and continuous variables. Their interest lies in the possibility of obtaining interesting continuous relaxations. The main advantage of the big M method (approach 3) is that it brings only linear inequality constraints, but the value of the M term needs to be chosen carefully since it shows a great deal of practical influence on the solver performance. Logical implications (approach 4) have the advantage of avoiding these types of problems, as they do not rely on a separate constant value. However, they tend to have weaker relaxations, a condition which may lead to longer solve times in a model. In this paper we will use the third approach for our implementation. Outlier detection also involves the ℓ_0 norm function to count the number of outliers. This counting function can be represented by introducing n binary variables $t_i \in \{0,1\}$ such as

$$\|\tau\|_0 = \sum_{i=1}^{n} t_i \quad \text{and} \quad t_i = 0 \Leftrightarrow \tau_i = 0, \ (x_i, y_i) \text{ is not an outlier.}$$

4.2.2 MIO Formulation of Problem MIO Formulation of Problem (4.4)

Introducing binary variables for both variables and outliers with two big M constraints, given appropriate parameters k_v, k_o, M_v and M_o, Problem (4.4) becomes for some $q \in$

{1,2}:

$$\begin{array}{lll} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n,z\in\{0,1\}^p,t\in\{0,1\}^n} & \frac{1}{2}\|X\beta+\tau-y\|_2^2 & \\ \text{s.t.} & \sum\limits_{j=1}^{p} z_j \le k_v \quad \text{and} \quad |\beta_j| \le z_j M_v, & j=1,\dots,p \\ & \sum\limits_{i=1}^{n} t_i \le k_o \quad \text{and} \quad |\tau_i| \le t_i M_o & i=1,\dots,n. \end{array} \tag{4.6}$$

This problem is a mixed binary quadratic program.

4.2.3 MIO Formulation of Problem (4.5)

To deal with the ℓ_1-norm term in Problem (4.5), we introduce two variables $\eta^+, \eta^- \in \mathbb{R}^p$ such that $|\beta_j = \eta_j^+ + \eta_j^-|$ and $\beta_j = \eta_j^+ - \eta_j^-$.

$$\begin{array}{lll} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n,\eta^+\in\mathbb{R}^p,\eta^-\in\mathbb{R}^p,z\in\{0,1\}^p,t\in\{0,1\}^n} & \frac{1}{2}\|X\beta+\tau-y\|_2^2+\lambda\sum\limits_{j=1}^{p}(\eta^+ + \eta^-) & \\ \text{s.t.} & \sum\limits_{j=1}^{p} z_j \le k_v \quad \text{and} \quad |\beta_j| \le z_j M_v, & j=1,\dots,p \\ & \sum\limits_{i=1}^{n} t_i \le k_o \quad \text{and} \quad |\tau_i| \le t_i M_o & i=1,\dots,n \\ & \eta_j^+ - \eta_j^- = \beta_j & j=1,\dots,p. \end{array} \tag{4.7}$$

4.3 Discrete First Order Algorithms

In this section we propose discrete first order methods to obtain near optimal solutions for Problems (4.4) and (4.5). These first order methods are used as warm starts to the MIO formulations leading to a significant decrease in computational time [BKM15]. Furthermore, as it will be shown in numerical experiments, the first order algorithms provide high quality solutions in a short time (compared to MIO problems). The proposed algorithms borrow ideas from alternating minimization and projected gradient descent methods.

In [BST14], a proximal alternating linearized minimization (PALM) algorithm for nonconvex and non-smooth problems was proposed to solve, under some assumptions, problems of the form:

$$\min_{x,y} \Psi(x,y) := f(x) + g(y) + H(x,y).$$

To deal with cardinality constraints, it is appropriate to introduce the problem of finding the projection of a vector $u \in \mathbb{R}^p$ onto the set of $k \le p$ sparse vectors

$$\begin{array}{ll} \min\limits_{v\in\mathbb{R}^p} & \frac{1}{2}\|v-u\|^2 \\ \text{s.t.} & \|v\|_0 \le k. \end{array} \tag{4.8}$$

This problem is easy and its solution $v^\star$ is given by sorting on the absolute value of vector $|u|$, that is by a sequence of indices (j) such that $|u_{(1)}| \ge |u_{(2)}| \ge \dots |u_{(j)}| \ge \dots \ge |u_{(p)}|$. Using these indices, the projection $v^\star = P_k(u)$ of u is the vector u itself with its smallest coefficients set to 0 that is

$$v^\star = P_k(u) = \begin{cases} u_j & \text{if } j \in \{(1),\dots,(k)\} \\ 0 & \text{else.} \end{cases}$$

We propose to use this projection mechanism, on both β and τ, to get a local solution to the initial Problems (4.4) and (4.5) at a low computational cost.

4.3.1 Discrete First Order Algorithm for Problem (4.4)

We use $f(\beta,\tau)$ to denote the objective function in (4.4):

$$\min_{\beta,\tau} \ F(\beta,\tau) := f(\beta,\tau) \ \text{ s.t } \ ||\beta||_0 \le k_v \ \ ||\tau||_0 \le k_o. \tag{4.9}$$

The partial derivatives of f are given by:

$$\nabla_\beta f(\beta,\tau) = X^t(X\beta+\tau-y) = G_\beta(\beta),$$

and

$$\nabla_\tau f(\beta,\tau) = (X\beta+\tau-y) = G_\tau(\tau).$$

By simple calculation, it can be shown that G_β and G_τ are Lipschitz with modules $L_1 = \sigma^2_{\max}(X)$ and $L_2 = 1$ respectively, where $\sigma_{\max}(X)$ is the maximum singular value of X.

We want to obtain local solutions for Problems (4.4) and (4.5) at a low computational cost. A possible way to achieve this goal consists of using the so-called block Gauss-Seidel iteration scheme on variables β and τ, also known as alternating minimization. To this end, a sequence $\{(\beta^\ell,\tau^\ell)\}_\ell \in \mathbb{N}$ is generated starting from some (β^0,τ^0) using the following scheme:

$$\begin{cases} \beta^{\ell+1} = arg\min\limits_{\beta\in\mathbb{R}^p} \ (\beta-\beta^\ell)^t X^t(X\beta^\ell+\tau^\ell-y) \\ \qquad\qquad \text{s.t.} \quad \|\beta\|_0 \le k_v \\ \qquad\qquad\quad \|\beta-\beta^\ell\|^2 \le d_v \end{cases} \quad \begin{cases} \tau^{\ell+1} = arg\min\limits_{\tau\in\mathbb{R}^n} \ (\tau-\tau^\ell)^t(X\beta^{\ell+1}+\tau^\ell-y) \\ \qquad\qquad \text{s.t.} \quad \|\tau\|_0 \le k_o \\ \qquad\qquad\quad \|\tau-\tau^\ell\|^2 \le d_o. \end{cases}$$

Where d_v and d_o are two given positive parameters that can be changed each step. The idea of the proximal method is, at each iteration, to minimize a regularized first-order approximation of the cost that can be interpreted as a local trust region mechanism [**?**, for details see for instance]]parikh2014proximal. This surrogate loss is also a local upper bound of the targeted loss since, for well chosen ρ_v and ρ_o, the Lagrange multipliers associated with the trust region constraints are

$$\begin{cases} \frac{1}{2}\|X\beta+\tau^\ell-y\|^2 \le \frac{1}{2}\|X\beta^\ell+\tau^\ell-y\|^2 + (\beta-\beta^\ell)^tX^t(X\beta^\ell+\tau^\ell-y) + \dfrac{1}{2\rho_v}\|\beta-\beta^\ell\|^2 \\ \frac{1}{2}\|X\beta^{\ell+1}+\tau-y\|^2 \le \frac{1}{2}\|X\beta^{\ell+1}+\tau^\ell-y\|^2 + (\tau-\tau^\ell)^t(X\beta^{\ell+1}+\tau^\ell-y) + \dfrac{1}{2\rho_o}\|\tau-\tau^\ell\|^2. \end{cases}$$

For each iteration, this method introduced by [BST14] and called the proximal alternating linearized minimization (PALM) algorithm, consists of minimizing the upper bounds as follows:

$$\begin{cases} \beta^{\ell+1} = arg\min\limits_{\beta\in\mathbb{R}^p,\|\beta\|_0\le k_v} (\beta-\beta^\ell)^tX^t(X\beta^\ell+\tau^\ell-y) + \dfrac{1}{2\rho_v}\|\beta-\beta^\ell\|^2 \\ \tau^{\ell+1} = arg\min\limits_{\tau\in\mathbb{R}^n,\|\tau\|_0\le k_o} (\tau-\tau^\ell)^t(X\beta^{\ell+1}+\tau^\ell-y) + \dfrac{1}{2\rho_o}\|\tau-\tau^\ell\|^2. \end{cases}$$

That is, after some algebra,

$$\begin{cases} \beta^{\ell+1} = arg\min\limits_{\beta\in\mathbb{R}^p,\|\beta\|_0\le k_v} \ \frac{1}{2}\|\beta-\beta^\ell+\rho_vX^t(X\beta^\ell+\tau^\ell-y)\|^2 \\ \tau^{\ell+1} = arg\min\limits_{\tau\in\mathbb{R}^n,\|\tau\|_0\le k_o} \ \frac{1}{2}\|\tau-\tau^\ell+\rho_o(X\beta^{\ell+1}+\tau^\ell-y)\|^2. \end{cases}$$

These two minimization problems are of the same kind as Problem (4.8) and thus the sequence can be generated by using two ℓ_0 projected gradient, that is:

$$\begin{cases} \beta^{\ell+1} = P_{k_\nu}\left(\beta^\ell - \rho_\nu X^t(X\beta^\ell + \tau^\ell - y)\right) \\ \tau^{\ell+1} = P_{k_o}\left(\tau^\ell - \rho_o(X\beta^{\ell+1} + \tau^\ell - y)\right). \end{cases}$$

Algorithm 4.1 presents the pseudo code of the PALM algorithm.

Algorithme 4.1 : Proximal alternating linearized minimization (PALM) [BST14].

Data : X, y, ε initialization $\beta, \tau = 0$
Result : β, τ
set $\rho_\nu \leq \frac{1}{\sigma_M^2}$ and $\rho_o \leq 1$
while *it has not converged* $(||\beta_{n+1} - \beta_n||_2 > \varepsilon)$ **do**
  $d \leftarrow \beta - \rho_\nu X^\top(X\beta + \tau - y)$ — variable selection
  $\beta \leftarrow P_{k_\nu}(d)$
  $\delta \leftarrow \tau - \rho_o(X\beta + \tau - y)$ — eliminating outliers
  $\tau \leftarrow P_{k_o}(\delta)$

This algorithm converges towards a local minima of Problem (4.4) since it fulfills the assumption needed for Theorem 3.1 in [BST14]. Indeed, the partial gradients $G_\beta(\beta) = X^\top(X\beta + \tau - y)$ and $G_\tau(\tau) = (X\beta + \tau - y)$ are globally Lipschitz with module respectively $\frac{1}{\sigma_M^2}$ and 1, σ_M being the largest singular value of X. Also, to prove the convergence, the stepsizes have to be chosen such that $\rho_\nu \leq \frac{1}{\sigma_M^2}$ and $\rho_o \leq 1$.

4.3.2 Discrete First Order Algorithm for Problem (4.5)

The main difference between Problems (4.4) and (4.5) is the presence of an ℓ_1 penalty in the objective function of (4.5). Let

$$\begin{aligned} F(\beta) &= \tfrac{1}{2}||X\beta + \tau - y||_2^2 + \lambda||\beta||_1 \ \text{ s.t } ||\beta||_0 \leq k_\nu \\ &= f(\beta) + \lambda||\beta||_1 \ \text{ s.t } ||\beta||_0 \leq k_\nu. \end{aligned}$$

The gradient of $f(\beta)$ is Lipschitz and continuous with parameter L_ν, in fact:

$$||\nabla f(\beta) - \nabla f(\tilde{\beta})||_2 \leq L_\nu||\beta - \tilde{\beta}||_2 \ \ \forall \beta, \tilde{\beta} \in \mathbb{R}^p.$$

whith $L_\nu = \sigma_{max}^2(X)$, where $\sigma_{max}(X)$ is the maximum singular value of X. Consequently, for $L \geq L_\nu$, we have the following upper bound to $f(\beta)$:

$$f(\beta) \leq f(\tilde{\beta}) + \langle \nabla f(\tilde{\beta}), \beta - \tilde{\beta}\rangle + \frac{L}{2}||\beta - \tilde{\beta}||_2^2 := Q_L(\beta, \tilde{\beta}) \ \ \forall \beta, \tilde{\beta} \in \mathbb{R}^p \, .$$

Our goal is to minimize this upper bound, that is:

$$\min_{||\beta||_0 \leq k_\nu} Q_L(\beta, \tilde{\beta}) + \lambda||\beta||_1 \Leftrightarrow \min_{||\beta||_0 \leq k_\nu} \frac{L}{2}||\beta - (\tilde{\beta} - \frac{1}{L}\nabla f(\tilde{\beta}))||_2^2 + \lambda||\beta||_1.$$

Dealing with the ℓ_1 penalty term necessitates the use of the soft-thresholding operator, while dealing with the cardinality constraint requires using the hard-thresholding operator (the projection operator).

To solve our problem, we introduce a thresholding operator [MRD17] combining both ideas:

$$S(u; k_\nu; \lambda) = arg \min_{||\beta||_0 \leq k_\nu} \frac{1}{2}||\beta - u||_2^2 + \lambda||\beta||_1 \, .$$

This threshold operator has the form:

$$\beta_i = \begin{cases} sign(u_i)max\{0, |u_i - \lambda|\} & i \in \{(1), (2), \ldots, (k_v)\} \\ 0 & \text{otherwise,} \end{cases}$$

where $(1), \ldots, (p)$ is a permutation of the indices $1, \ldots, p$.

Summing up all together, we obtain that the β update is given by: By introducing the soft-thresholding operator, it can be shown that the $\beta-$ update is given by:

$$\begin{aligned} \beta &\leftarrow \beta - \rho_v X^t(X\beta + \tau - y) \\ \beta &\leftarrow sign(\beta)\max(0, |\beta| - \rho_v\lambda) \\ \beta &\leftarrow P_{k_v}(\beta). \end{aligned}$$

Putting all that together leads to Algorithm 4.2.

Algorithme 4.2 : Proximal alternating linearized minimization (PALM) for Problem (4.5).

Data : X, y initialization $\beta, \tau = 0$
Result : β, τ
set $\rho_v \leq \frac{1}{\sigma_M^2}$ and $\rho_o \leq 1$
while *it has not converged* ($||\beta_{n+1} - \beta_n||_2 > 10^{-6}$) **do**
  $d \leftarrow \beta - \rho_v X^\top (X\beta + \tau - y)$
  $d \leftarrow sign(d)\big(max(0, |d| - \rho_v\lambda)\big)$
  $\beta \leftarrow P_{k_v}(d)$
  $\delta \leftarrow \tau - \rho_o(X\beta + \tau - y)$
  $\tau \leftarrow P_{k_o}(\delta)$

Note that this algorithm converges towards a local minima of Problem (4.5) since it fulfills the assumption needed for Theorem 3.1 in [BST14].
In the rest of the paper, algorithm (4.1) will be denoted by $PALM_0$, while algorithm (4.2) will be denoted by $PALM_{0,1}$

4.4 Experiments on Synthetic Data Sets

The aim of the experiments is to shed the light on the superior performance of Problem (4.5) over Problem (4.4), especially for low SNR values. We note that, Problem (5) will be denoted by $L_{0,1}$, Problem (4) will be denoted by L_0, Algorithm (4.2) will be denoted by $PALM_{0,1}$ and Algorithm (4.1) will be denoted by $PALM_0$.

4.4.1 Setup

Given n, p (problem dimensions), s (sparsity level), beta-type (pattern of sparsity), ρ (predictor auto-correlation level), and ν (SNR level) as in [HTT], the setup can be described as follows:

— We define coefficients $\beta_0 \in \mathbb{R}^p$ with a sparsity level $s = 5$ and with beta-type, as described below.

— We draw the rows of the matrix $X \in \mathbb{R}^{n\times p}$ from $N_p(0, \Sigma)$, where $\Sigma \in \mathbb{R}^{p\times p}$ has entry (i, j) equal to $\rho^{|i-j|}$. We considered two values $\rho = 0.3$ and $\rho = 0.6$

— The columns of X are standardized to have mean zero and unit ℓ_2 norm.

— We then draw the vector $y \in \mathbb{R}^n$ from $\mathrm{N}_n(\mathrm{X}\beta_0, \sigma^2\mathrm{I})$, with σ^2 defined to meet the desired SNR level, i.e., $\sigma^2 = \beta_0^{\mathrm{T}}\Sigma\beta_0/\nu$. We considered three values of SNR 0.5, 1 and 3.

— The data set (X, y) is used as a training set. A validation set (X_t, y_t) is created in the same way as (X, y) with $n_t = 1000$.

— 5% of outliers were added to the training data set by following a normal $\mathrm{N}(50, \sigma)$ instead of $\mathrm{N}(0, \sigma)$.

— In order to determine the values of k_v, M_v, k_o, M_o and λ, we run the PALM algorithms for and pick the solution minimizing the error on the testing data, let β_{PALM} denote this solution.

— $\mathrm{M}_v = (1+\alpha)||\beta_{palm}||_\infty$, $\mathrm{M}_o = (1+\alpha)||\tau_{palm}||_\infty$ with $\alpha = 0.1$, k_v and k_o are set as the number of nonzero elements in the solutions β_{PALM} and τ_{PALM} respectively.

— We considered three configurations: the low setting with $n = 10, p = 20$, the medium setting $n = 50, p = 100$ and the high-dimensional setting $n = 100, p = 1000$.

Coefficients: We considered two settings for the coefficients $\beta_0 \in \mathbb{R}^p$:

— beta-type 1: $\beta_0 = (1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ \underbrace{0,\ldots,0}_{p-10 \text{ times}})$;

— beta-type 2: β_0 has its first 5 components equal to 1, and the rest equal to 0;

4.4.2 Selecting Tuning Parameters

In order to compare Problem (4.4) (L_0), Problem (4.5) ($\mathrm{L}_{0,1}$), Algorithm (4.1) (PALM_0) and Algorithm (4.2) ($\mathrm{PALM}_{0,1}$), we considered two different experiments:

Experiment 1:

To shed the light on the influence of the ℓ_1 penalty, we run both PALM algorithms (4.1) and (4.2) on the data set (X, y) for $k_v \in \{2, 5, 10, 15\}$ and for $k_o = 5\%$ that is the true percentage of outliers added to the data. In order to determine λ, we run PALM algorithm (4.2) for 100 values of λ starting with $\lambda_1 = ||\mathrm{X}'y||_\infty$. $\{\lambda_i\}_{i=1}^{100}$ is a geometrically spaced sequence with $\lambda_{100} = 10^{-4}\lambda_1$.

We pick the solutions $\hat{\beta}_{\mathrm{PALM}_{0,1}}(\lambda, k_v)$ and $\hat{\beta}_{\mathrm{PALM}_0}(k_v)$ with smallest error $||\mathrm{X}_t\beta - y_t||_2^2$. These two solutions are used as warm starts for MIO formulations.

We report the prediction error defined by:

$$\mathrm{PE} = ||\mathrm{X}\beta - \mathrm{X}\beta_0||_2.$$

The results have been averaged over ten different replications of data.

Experiment 2 on Synthetic Data

In this experiment, we want to mimic real word setting, k_v and k_o are not fixed anymore. To this end we take a 3D grid of tuning parameters. $\{\lambda_i\}_1^{\mathrm{N}}$ is a geometrically spaced sequence of 100 values, $k_v \in \{1, \ldots, 15\}$ and $k_o \in \{0, 0.025n, 0.05n\}$, $\lambda_1 = ||\mathrm{X}^t y||_\infty$ and $\lambda_{\mathrm{N}} = 10^{-4}\lambda_1$. We train Algorithm (4.2) on (X, y) and find $\hat{\beta}(\lambda, k_v, k_o)$ with minimal error on (X_t, y_t). This solution $\hat{\beta}(\lambda, k_v, k_o)$ is used as a warm start for the MIO formulation (4.5). The same is done for Problem (4.4) but without λ.

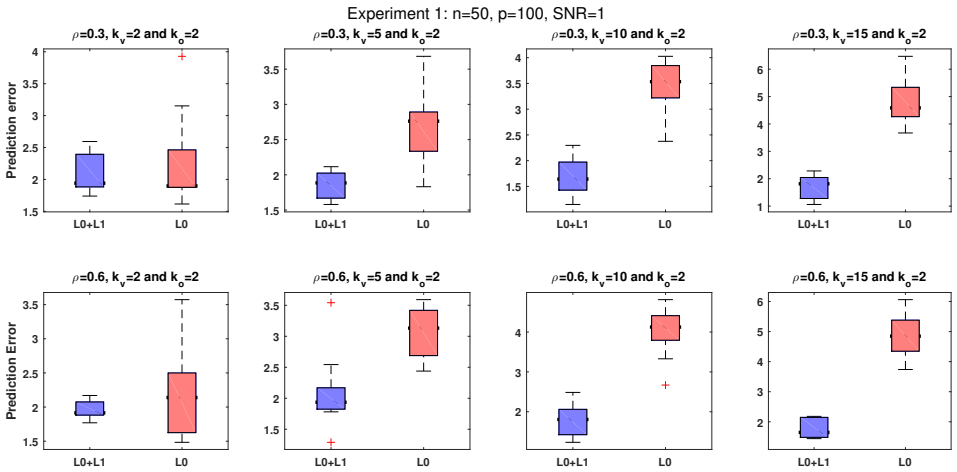

FIGURE 4.1 – Experiment 1 showing the effect of the ℓ_1 penalty.

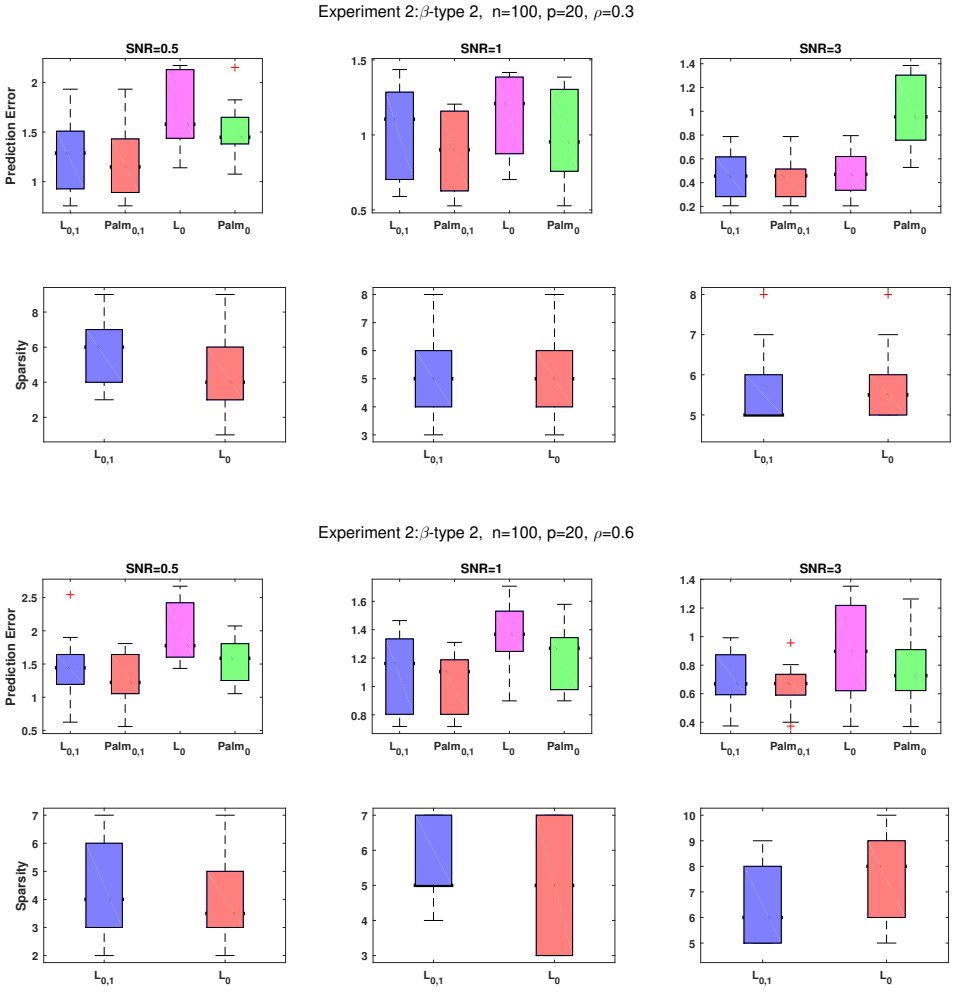

FIGURE 4.2 – Experiment 2 β-type 2 simulations with $n = 100$ and $p = 20$. The top two rows display the results for ρ = 0.3 while the bottom two rows display the results for ρ = 0.6. Prediction error refers to the best predictive models obtained after tuning on separate validation set. Sparsity refers to the corresponding number of nonzero coefficients. Three SNR values were considered.

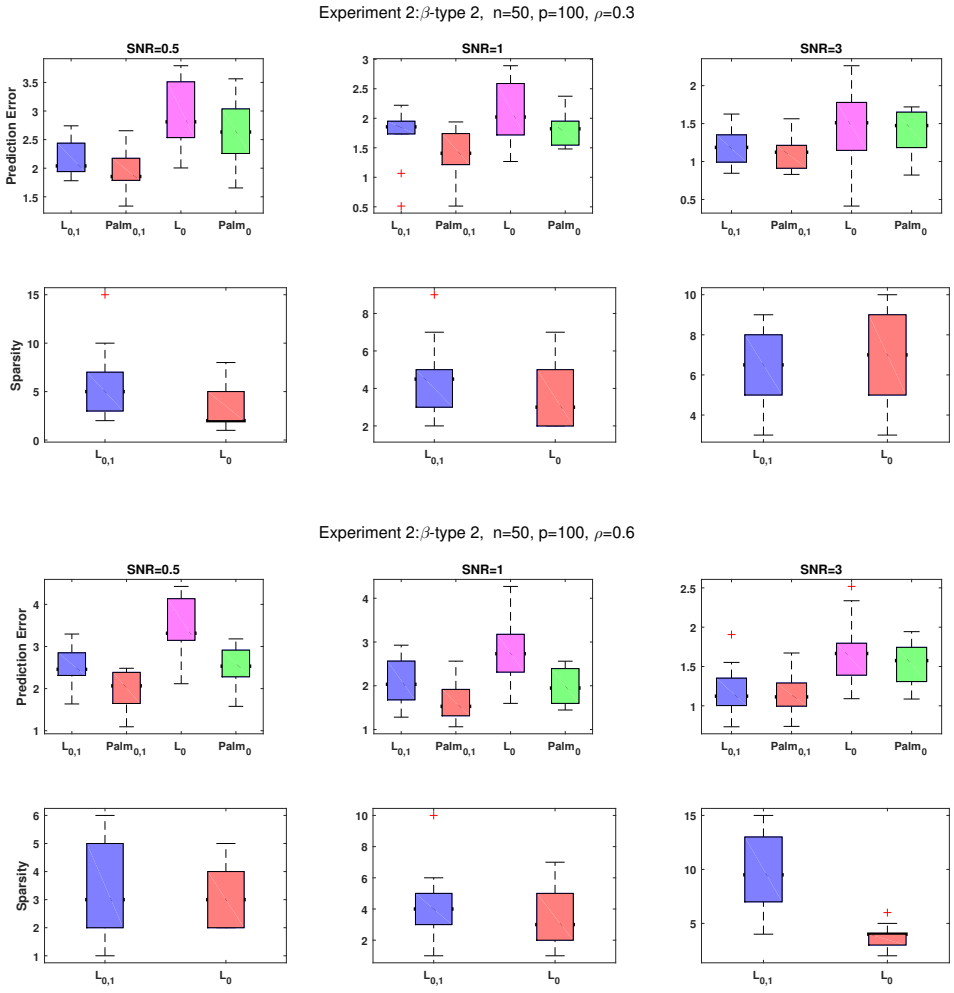

FIGURE 4.3 – Experiment 2 β-type 2 simulations with $n = 50$ and $p = 100$. The top two rows display the results for ρ = 0.3 while the bottom two rows display the results for ρ = 0.6. Prediction error refers to the best predictive models obtained after tuning on separate validation set. Sparsity refers to the corresponding number of nonzero coefficients. Three SNR values were considered.

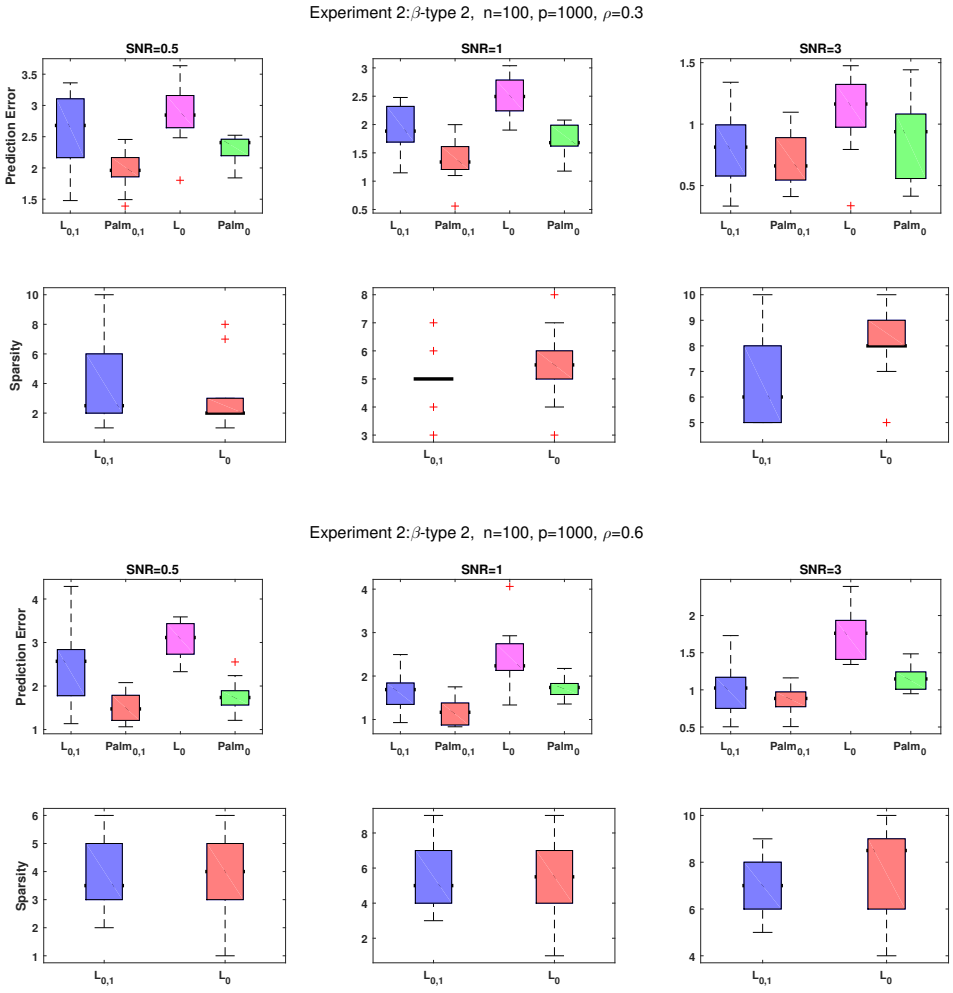

FIGURE 4.4 – Experiment 2 β-type 2 simulations with $n = 100$ and $p = 1000$. The top two rows display the results for ρ = 0.3 while the bottom two rows display the results for ρ = 0.6. Prediction error refers to the best predictive models obtained after tuning on separate validation set. Sparsity refers to the corresponding number of nonzero coefficients. Three SNR values were considered.

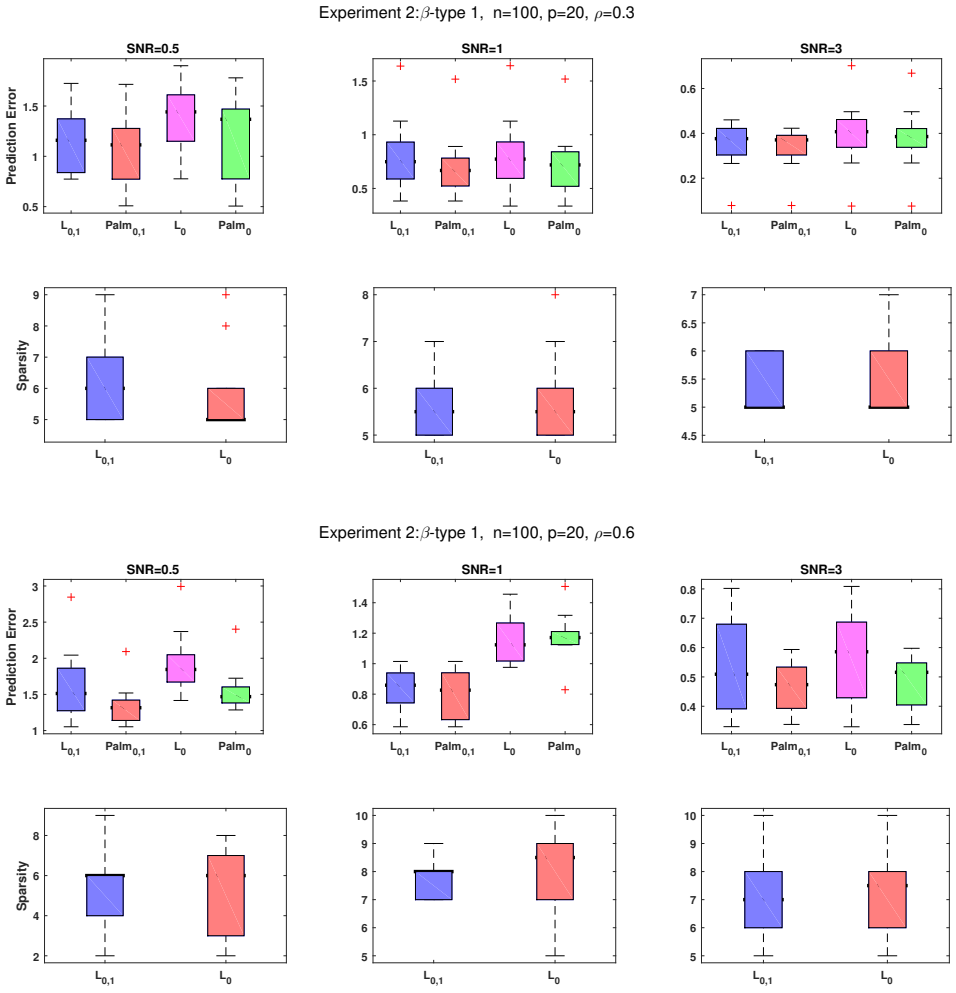

FIGURE 4.5 – Experiment 2 β-type 1 simulations with $n = 100$ and $p = 20$. The top two rows display the results for ρ = 0.3 while the bottom two rows display the results for ρ = 0.6. Prediction error refers to the best predictive models obtained after tuning on separate validation set. Sparsity refers to the corresponding number of nonzero coefficients. Three SNR values were considered.

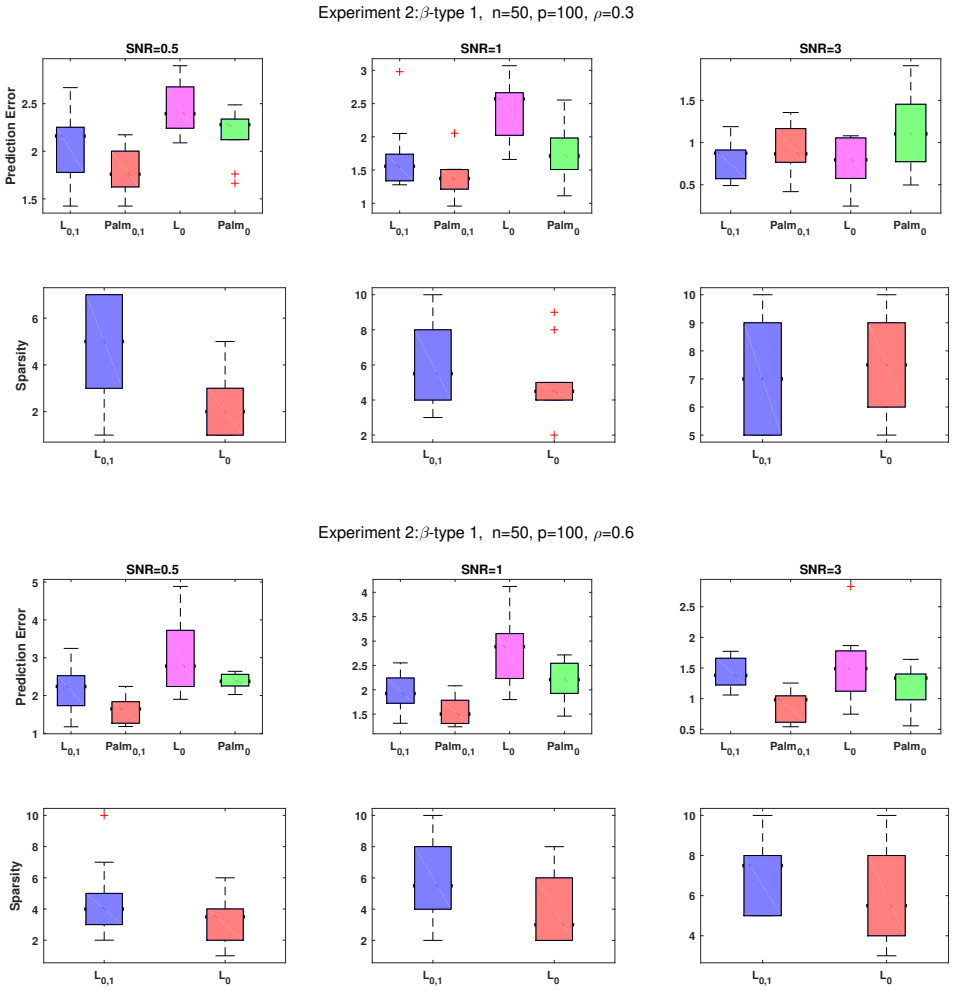

FIGURE 4.6 – Experiment 2 β-type 1 simulations with $n = 50$ and $p = 100$. The top two rows display the results for ρ = 0.3 while the bottom two rows display the results for ρ = 0.6. Prediction error refers to the best predictive models obtained after tuning on separate validation set. Sparsity refers to the corresponding number of nonzero coefficients. Three SNR values were considered.

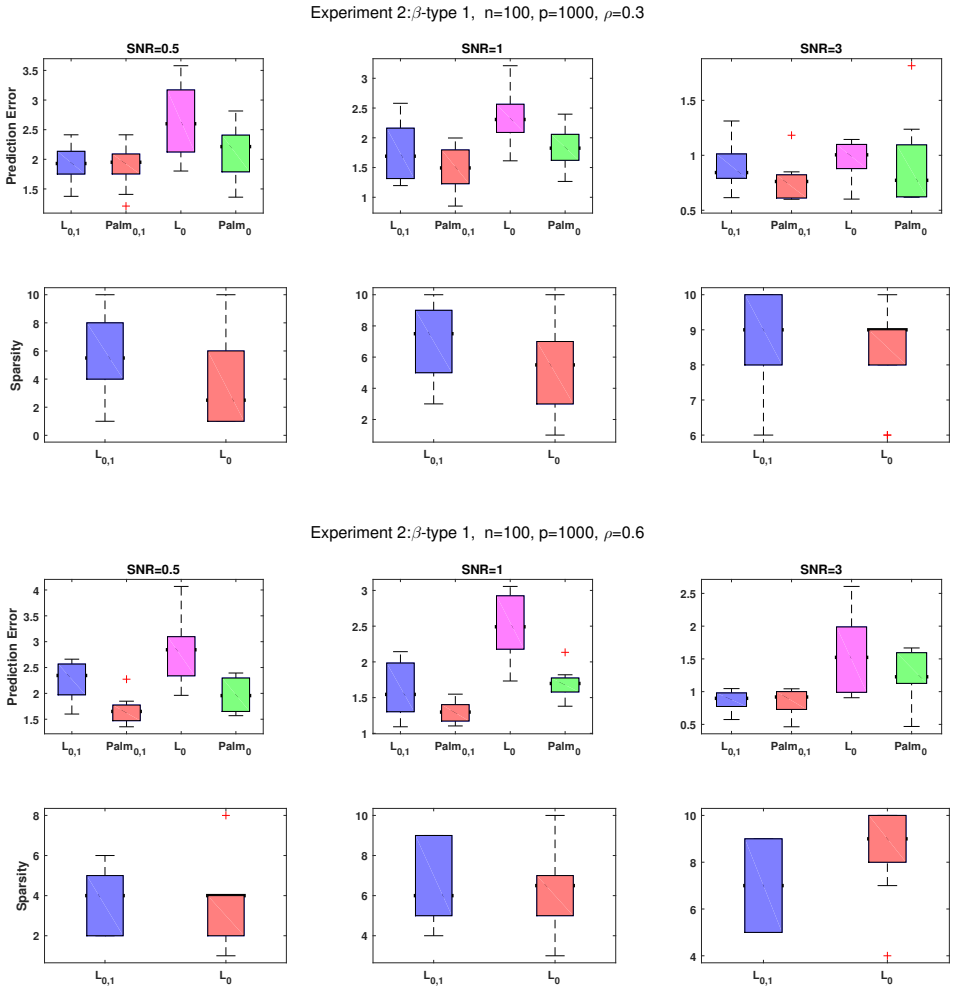

FIGURE 4.7 – Experiment 2 β-type 1 simulations with $n = 100$ and $p = 1000$. The top two rows display the results for ρ = 0.3 while the bottom two rows display the results for ρ = 0.6. Prediction error refers to the best predictive models obtained after tuning on separate validation set. Sparsity refers to the corresponding number of nonzero coefficients. Three SNR values were considered.

4.4.3 Computational Time

The $\text{PALM}_{0,1}$ estimator is tuned over 100 values of λ, 15 values of k_v and 3 values of k_o. We denote by $\beta_{\text{PALM}_{0,1}}(\lambda, k_v, k_o)$ the 3D family of solutions obtained. For $n = 100$, $p = 20$ it takes about 15 seconds to compute this family. For $n = 50$, $p = 100$ computing the family takes about 1 minute. For $n = 100$, $p = 1000$ about 10 minutes were needed to find this family. The threshold of the discrete first order algorithms is 10^{-6}. Once the family is obtained, the best solution $(\hat{\lambda}, \hat{k_v}, \hat{k_o})$ on a separate validation set. This solution is used as a warm start for the MIO solver with a time limit of 1000 seconds which seemed to be enough to obtain a global solution for the majority of problems solved. However even if the optimal solution is not certified, a near optimal solution with a high quality is guaranteed. We report the two solutions obtained by the DFO algorithm and the MIO formulation and we denote them by $\text{PALM}_{0,1}$ and $\text{L}_{0,1}$ respectively. The same is done to obtain the solutions PALM_0 and L_0 representing the solutions obtained by Algorithm (4.1) and Problem (4.4).

4.4.4 Results

We explore the properties of our estimator numerically on synthetic datasets with varying values of n, p, SNR and correlations among the predictors.

The experimental results are summarized below:

— In Figure (4.1), it is shown that as k_v gets greater than the true sparsity level, the prediction error of L_0 decreases. However, the performance of $\text{L}_{0,1}$ stay stable, which demonstrate the significant importance of using the ℓ_1 penalty term since in practice, if the PALM algorithms are used to tune the parameters by a k-fold cross validation, we remarked that they overestimate the sparsity level especially for low SNR values. Thus using this penalty term, will ensure avoiding the decrease of the performance for overestimated sparsity level s.

— For $n = 100$, $p = 20$ Figures (4.2) and (4.5) show that when SNR=0.5, $\text{PALM}_{0,1}$ produces the best solution in terms of prediction error. $\text{L}_{0,1}$ is close to $\text{PALM}_{0,1}$ and both outperform L_0 and PALM_0. As the SNR value increases, the difference between the penalized problem and the L_0 problem decreases until obtaining almost similar results for SNR=3. Furthermore, the poor performance of the L_0 problem are noticed for low SNR values because of the overfitting effect. We also note that as ρ increases, the ℓ_1 penalty ensures obtaining better solutions.
In terms of sparsity, the L_0 produced a little bit sparser solutions.

— For the medium setting $n = 50$ and $p = 100$, we can see that in Figures (4.3) and (4.6), the penalized problems $\text{L}_{0,1}$ and $\text{PALM}_{0,1}$ still perform better than L_0 and PALM_0 but now with a bigger difference.

— For the high dimensional setting $n = 100$ and $p = 1000$, same results are obtained but in this case PALM algorithms have an advantage over their corresponding MIO formulations. This may be explained by the fact the MIO formulations have a time limit of 1000 seconds.

In general, the ℓ_1 penalty term improved the ℓ_0 formulation for low SNR values, higher correlation ρ values and high dimensional settings. The added shrinkage fixes the overfitting behavior of L_0. Finally, the good performance of of both PALM algorithms is clearly noticed.

TABLE 4.1 – Summary of used datasets.

Name of the dataset	*number of instances n*	*number of attributes p*	*Origin*
Auto Mpg	398	8	UCI
Concrete Compressive Strength	1030	9	UCI
Concrete Slump Test	103	10	UCI
Diabetes	442	10	`stat.ncsu.edu`
Forest Fires	517	13	UCI

TABLE 4.2 – Mean Squared Error Rates (Standard Deviations) of $L_{0,1}$, $PALM_{0,1}$, L_0 and $PALM_0$ on Five Real Data Sets.

	$L_{0,1}$	$PALM_{0,1}$	L_0	$PALM_0$
Auto MPG	**0.4332 (0.0472)**	0.4379 (0.0483)	0.4436 (0.0476)	0.4365 (0.0477)
Concrete Compressive Strength	0.7366 (0.0342)	0.7465 (0.0347)	**0.7345 (0.0334)**	0.7444 (0.0331)
Concrete Slump Test	0.6015 (0.1136)	**0.5925 (0.1228)**	0.6058 (0.1100)	0.6022 (0.1344)
Diabetes	5.9346 (0.3330)	5.9485 (0.3459)	5.9146 (0.2996)	**5.9133 (0.2962)**
Forest Fires	**5.6359 (3.9369)**	5.6372 (3.9370)	5.6456 (3.9360)	5.6396 (3.9385)

4.5 Experiments on Real Data Sets

We compare the performance of all mentioned methods above on five real data sets from the UCI Machine Learning Repository presented in Table 4.1. We split each data set into three parts: the training set (40%), the validation set (40%) and the testing set (20%). The training set was used to train both proximal algorithms for a variety of combinations of input parameters. For each combination of parameters, the mean squared error on the validation set was calculated, and this was used to select the best combination of parameters for $PALM_{0,1}$ and $PALM_0$. Finally, all methods were trained by using these best parameters on the combined training and validation sets, before reporting the out-of-sample prediction error, defined below, on the testing set. We note that the columns of X were standardized to have zero means and unit ℓ_2-norms and that the response vector y was standardized to have zero mean. All methods were trained, validated, and tested on the same random splits, and computational experiments were repeated ten times for each data set with different splits. For each data set and regression method, we report the average of the results and the standard deviation in parentheses for the raw data.

$$\text{Prediction Error}: \ \text{PE} = \frac{1}{n_t}||\mathrm{X}_{test}\beta - y_{test}||_2 \ ,$$

where n_t represents the number of observations of X_{test}. Results in Table (4.2) sheds the light on the good performance of the proposed algorithm. In fact, over five data sets, the ℓ_1 regularized methods performs better than ℓ_0 methods on three data sets. However there is no big difference between results on all methods. This can be explained by the fact that for high SNR values these methods perform almost similarly and the ℓ_1 regularized problem fixes the overfitting problem of the ℓ_0 problems for low SNR values.

4.6 Conclusion

In this chapter, an ℓ_1 regularized method performing both feature selection and outliers detection in linear regression. The problem is formulated as a mixed integer optimization problem and warm started by a discrete first order algorithm. We present an empirical comparison between the ℓ_0 algorithms and their ℓ_1 cousins. Results on synthetic data sets show that the regularized problems overperformed the ℓ_0 methods for low SNR

values and as the SNR increases, these methods become almost similar. Furthermore, results on real data sets show that the proposed method produce high quality solutions. Based on the above we propose to use the proximal algorithm (4.2) in practical use if we are not interested in obtaining the global solution, since it performs well for both high and low SNR values in a short time and it can be used for high dimensional problems.

Chapitre 5

Support Vector Machines

Sommaire

5.1 Introduction

Data classification sets the basic steps for anticipating corresponding labels of new points on the basis of a pre-defined set of labeled training points specially in the domain of machine learning and statistics. It is considered to be one of the supervised learning problems and it is applied in various domains such as document classification, handwriting recognition, internet search engines, etc. Instance-based learning methods, neural networks, decision trees, support vector machines (SVM) and many other methods have been developed to deal with data classification problems.

The Support Vector Machines often translated by the name of Large Margin Separator (SVM) are a class of learning algorithms initially defined for discrimination, i.e. the prediction of a binary qualitative variable. In the case of the discrimination of a dichotomous variable, they are based on the search for the optimal margin hyperplane which, when it is possible, correctly classifies or separates the data while being as far away as possible from all observations. The principle is therefore to find a classifier, or a discrimination function, whose generalization capability (forecasting quality) is as high as possible.

This approach stems directly from Vapnik's work in learning theory from 1995 onwards [CV95]. It focused on the generalization (or prediction) properties of a model by controlling its complexity. The principle founder of SVMs is precisely to include in the estimation the control of the complexity, i.e. the number of parameters which is associated in this case with the number of support vectors. Vapnik's other guiding idea in this development is to avoid substituting the original objective of discrimination with a or

problems that are ultimately more complex to solve, such as e.g. the non-parametric estimation of the density of a multidimensional law in discriminant analysis.

The basic principle of SVMs is to reduce the problem of discrimination to the linear problem of finding an optimal hyperplane. Two ideas or tricks to achieve this goal:

— The first one consists in defining the hyperplane as the solution of a constrained optimization problem whose objective function is expressed only by scalar products between vectors and in which the number of "active" constraints or support vectors controls the complexity of the model.

— The search for non-linear separating surfaces is obtained by introducing a kernel function in the scalar product implicitly inducing a non-linear transformation of the data to a larger feature space. Hence the commonly encountered name kernel machine. On the theoretical level, the kernel function defines a Hilbertian space, called self-replicating and isometric by the nonlinear transformation of the initial space and in which the linear problem is solved.

This tool is becoming widely used in many types of applications and proves to be a serious competitor of the most efficient algorithms. The introduction of kernels, specifically adapted to a given problem, gives it a great flexibility to adapt to very diverse situations (pattern recognition, genomic sequence recognition, etc.), of characters, spam detection, diagnostics...). Note that, on the algorithmic level, these algorithms are more penalized by the number of observations, i.e. the number of potential support vectors, than by the number of variables. Nevertheless, high-performance versions of the algorithms enable to take into account large databases in times of acceptable calculations.

5.2 Linear Support Vector Machines

A discrimination problem is said to be linearly separable when there is a linear decision function (also called a linear separator), of the form $f(x) = sign(h(x))$ with $h(x) = w^{\mathrm{T}}x + b$, $w \in \mathbb{R}^p$, $b \in \mathbb{R}$, correctly classifying all observations in the learning set ($f(x_i) = y_i$, $i = 1, \ldots, n$). To any decision function and thus to linear decision functions we can associate a decision boundary:

$$\Delta(\mathrm{W}, b) = \{x \in \mathbb{R}^p \,|\, w^{\mathrm{T}}x + b = 0\}.$$

5.2.1 Basics: Hyperplane, Margin and Support Vectors

For two given classes, the goal of SVM is to find a classifier that will separate the data and maximize the distance between these two classes. With SVM, this classifier is linear and called **hyperplane**. The nearest points, which alone are used for determining the hyperplane, are called support vectors. There are many hyperplanes that separate the two classes of examples. The principle of the SVM is to choose the one that will maximize the minimum distance between the hyperplane and the training examples (i.e. the distance between the hyperplane and the support vectors), this distance is called the margin. Intuitively, having a wider margin provides more security when classifying a new example. Moreover, if we find the classifier that behaves best with respect to the learning data, it is clear that it will also be the one that will best classify the new examples.

5.2.2 SVM for Binary Classification

The generic problem of binary classification starts with an input domain $\mathrm{E} \subseteq \mathbb{R}^p$ to represent the values of the explanatory variables and a class domain $\mathrm{Y} = \{-1, 1\}$ to act as labels. Let $\mathrm{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a data set. Assuming linear separability, the goal in binary classification is to use the data to estimate the p-dimensional vector $\mathrm{W} = (w_1, \ldots, w_p)^{\mathrm{T}}$ and the constant b, so that the hyperplane

$$\mathrm{H}(w, b) = \{x : w^{\mathrm{T}} x + b = 0\},$$

separates the observations x_i into their class labels -1 and $+1$. Mathematically, the goal is to find a function $f : \mathrm{E} \longrightarrow \{-1, 1\}$ so that:

$$class(x_i) = f(x_i) = sign(w^{\mathrm{T}} x_i + b) = \begin{cases} +1 & w^{\mathrm{T}} x_i + b \geq 0 \\ -1 & w^{\mathrm{T}} x_i + b < 0. \end{cases}$$

The natural way to quantify the performance of a classifier is using the 0-1 loss function: If the true class of x is y and the classifier delivers $f(x)$, the loss incurred, or the misclassification error, is

$$\ell(y, f(x)) = \mathrm{I}(f(x) \neq y), \tag{5.1}$$

which is 0 if the classifier is correct and 1 otherwise, hence the name. However this loss function is non convex and non differentiable. To this end the hinge-loss, which is a convex surrogate for the 0-1 loss, was used.

Hard SVM

Maximizing the margin between two parallel hyperplanes that separate the two classes of data is called the Hard-SVM. Given $\mathrm{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$, the margin maximization principle applied to linear classifiers consists of finding the function $h(x) = w^{\mathrm{T}} x + b$ that achieves [CFZ09]:

$$\begin{aligned} &\max_{w,b} \Big(\min_{i, y_i = +1} \frac{|w^{\mathrm{T}} x_i + b|}{||w||} + \min_{i, y_i = -1} \frac{|w^{\mathrm{T}} x_i + b|}{||w||} \Big) \\ &\text{subject to } y_i (w^{\mathrm{T}} x_i + b) \geq 1, \qquad i = 1, \ldots, n. \end{aligned}$$

This optimization problem does not have a unique solution: if w and b are solutions, then for any constant $k \geq 1$, kw and kb are also solutions. Indeed, when $||kw|| \geq ||w||$, the boundaries of the classes move closer to the hyperplane $\mathrm{H}(w, b)$, contrary to the goal of large margins. It would be nice to have a unique solution that makes w and b as small as possible. To do this, one seeks the smallest $k \in (0, 1]$ such that kw and kb remain solutions to the constrained optimization problem. Since small values of k pushes the boundaries of the classes away from the separating hyperplane, the constraint $y_i (w^{\mathrm{T}} x_i + b) \geq 1$ must be enforced. So, solutions are characterized by the fact that at least one pair (x_i, y_i) exists such that $y_i (w^{\mathrm{T}} x_i + b) = 1$ holds for each of the two classes. This means that it is enough for the points to be correctly classified; how far within its class a point is does not matter. Now, since $y_i = \pm 1$, an optimal solution to the minimization of the distance is characterized by $w^{\mathrm{T}} x_i^\star + b = +1$ for some points $x_i^\star$ with $y_i^\star = +1$ and $w^{\mathrm{T}} x_i^\star + b = -1$ for some other points $x_i^\star$ with $y_i^\star = -1$. In this context, for given w and b, the canonical hyperplane for points of class $+1$ is defined by:

$$\mathrm{H}_{+1}(w, b) = \{x : w^{\mathrm{T}} x + b = +1, \text{ for } y = +1\},$$

and the canonical hyperplane for points of class -1 is defined by:

$$\mathrm{H}_{-1}(w,b) = \{x : w^{\mathrm{T}}x + b = -1, \text{ for } y = -1\}.$$

Using the results above, it can be shown that the margin is given by: $\mathrm{M} = \dfrac{2}{||w||}$. Now the optimization problem can be written as:

$$\begin{array}{ll} \max\limits_{w,b} & \dfrac{2}{||w||} \\ s.t & y_i(w^{\mathrm{T}}x + b) \geq 1, \qquad i = 1,\ldots,n \end{array}$$

The classical formulation of the SVM is obtained by minimizing $\dfrac{||w||}{2}$ instead of maximizing $\dfrac{2}{||w||}$. This is equivalent to minimizing $\dfrac{||w||^2}{2}$. Then the obtained optimization problem can be written as:

$$\begin{array}{ll} \min\limits_{w,b} & \dfrac{||w||_2^2}{2} \\ s.t & y_i(w^{\mathrm{T}}x + b) \geq 1, \qquad i = 1,\ldots,n \end{array}$$

This problem is a constrained quadratic optimization problem of the form:

$$\begin{array}{ll} \min\limits_{w,b} & \dfrac{1}{2}z^{\mathrm{T}}\mathrm{A}z - d^{\mathrm{T}}z \\ s.t & \mathrm{B}z \leq e\,, \end{array}$$

where $z = (w,b)^{\mathrm{T}} \in \mathbb{R}^{p+1}$, $d = (0,\ldots,0)^{\mathrm{T}} \in \mathbb{R}^{p+1}$, $\mathrm{A} = \begin{bmatrix} \mathrm{I} & 0 \\ 0 & 0 \end{bmatrix}$, with I the identity matrix of $\mathbb{R}^p$, $\mathrm{B} = -[diag(y)\mathrm{X}, y]$, $e = -(1,\ldots,1)^{\mathrm{T}} \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ and X is the observations $n \times p$ matrix. This problem is convex since the matrix A is semidefinite positive. It therefore admits a single solution (which exists since the problem is linearly separable by hypothesis) and the necessary conditions of first-order optimality are also sufficient. This (so-called primal) problem admits an equivalent dual formulation which is also a quadratic program. Solving the SVM problem on linearly separable data can be done directly (from the primal formulation) for example by using a Gauss-Seidel stochastic method, an active set method, an inner point algorithm, a Newton's algorithm with confidence region or a conjugated gradient type. However, it is interesting to go through the dual formation of this problem:

— The dual problem is a quadratic program of size n (equal to the number of observations) which may be easier to solve than the primal problem,

— the dual formulation reveals the Gram matrix $\mathrm{X}\mathrm{X}^{\mathrm{T}}$ which allows in the general (non-linear) case to introduce non-linearity through cores.

In order to explain the necessary conditions of optimality of the first order it is classical when dealing with an optimization problem under constraints to explain its Lagrangian. In the case of SVM it is written :

$$\mathscr{L}(w,b,\alpha) = \frac{1}{2}||w||_2^2 + \sum_{i=1}^{n} \alpha_i (y_i(w^{\mathrm{T}}x_i + b) - 1)\,,$$

where $\alpha_i \geq 0$ are the Lagrange multipliers associated with the constraints. The optimality conditions of Karush, Kuhn and Tucker (KKT) of the quadratic program associated with

SVM allow to characterize the solution of the primal problem $(w^\star, b^\star)$ and the associated Lagrange multipliers α by the following system of equations:

$$\begin{aligned}
\frac{\partial \mathscr{L}}{w} &= 0 \\
\frac{\partial \mathscr{L}}{b} &= 0 \\
(y_i(w^{\mathrm{T}} x_i + b) - 1) &\geq 0 \\
\alpha_i &\geq 0 \\
\alpha_i(y_i(w^{\mathrm{T}} x_i + b) - 1) &= 0,
\end{aligned}$$

which leads to:

$$\begin{aligned}
w - \sum_i \alpha_i y_i x_i &= 0 \\
\sum_i \alpha_i y_i &= 0 \\
(y_i(w^{\mathrm{T}} x_i + b) - 1) &\geq 0 \\
\alpha_i &\geq 0 \\
\alpha_i(y_i(w^{\mathrm{T}} x_i + b) - 1) &= 0,
\end{aligned}$$

5.3 Robust and Sparse Support Vector Machines

In support vector machine (SVM) classification, the natural way to quantify the performance of a classifier is via the 0-1 loss. This loss is non-convex and considered to be NP-hard. To this end, the hinge loss, which is convex, was introduced for the first time with [CV95]. Since then, it has become one of the most popular classifiers. An important reason behind the popularity of SVM is its significant empirical success in various applications such as data mining, engineering and bio-informatics [Lee10].
Considering training examples $x_i \in \mathbb{R}^p$ with their respective labels $y_i \in \{-1, 1\}, \quad i = 1, \ldots, n$. The main goal of SVM is to find a hyperplane (classifier) by introducing hard margins for separable data and soft margins for linearly non-separable data, the purpose of which is to separate data as far as possible from the hyperplane. A decision hyperplane can be defined by an intercept term b and a decision hyperplane normal vector w which is perpendicular to the hyperplane. This vector is commonly referred to, in the machine learning, literature as the weight vector. To choose among all the hyperplanes that are perpendicular to the normal vector, we specify the intercept term b. Because the hyperplane is perpendicular to the normal vector, all points x on the hyperplane satisfy $w^{\mathrm{T}} x + b = 0$. Let the margin be defined as the distance from the hyperplane to the closest point across both classes. It can be shown that the width of the margin is equal to $\frac{2}{||w||_2}$, thus maximizing this width is equivalent to minimizing the norm $||w||_2^2$ (or $\frac{1}{2}||w||_2^2$). To obtain the optimal hyperplane, one should solve the following optimization problem:

$$\begin{aligned}
\min_{w,\xi} \quad & \tfrac{1}{2}\|w\|_2^2 + \mathrm{C}\sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & y_i(w^{\mathrm{T}} x + b) \geq 1 - \xi_i \quad i = 1 \ldots n \\
& \xi_i \geq 0 \qquad\qquad\qquad\quad i = 1 \ldots n.
\end{aligned} \tag{5.2}$$

where ξ is a slack variable and C is a parameter controlling the trade-off between a large margin and a less constrained violation. The dual problem can be formulated through the

use of Lagrange multipliers:

$$\begin{aligned} \max_{\alpha} \quad & C\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}} x_j \\ \text{s.t} \quad & 0 \le \alpha_i \le C \qquad i = 1,\dots,n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0. \end{aligned}$$

Both the primal and dual are convex quadratic optimization problems. Because the dual problem has fewer decision variables, and the majority of these variables tend to be equal to zero, it is typically the problem solved in practice [FHT01].
While algorithmic advances in integer optimization combined with hardware improvements have resulted in an astonishing 200 billion factor speedup in solving Mixed Integer Optimization (MIO) problems [BKM16], this rapid development of MIO enabled [TLX+14] to reformulate the 0-1 loss classification problem as a mixed integer optimization problem and use it to solve small-scale classification problems.
In addition to all benefits listed above, SVM suffers from the existence of outliers and the existence of irrelevant features (especially for high dimensional data sets). Indeed, in the past three decades, the dimensionality of the data involved in machine learning and data mining tasks has increased explosively. Data with extremely high dimensionality has presented serious challenges to existing learning methods [FHT01, LM07]. With the presence of a large number of features, a learning model tends to overfit, resulting in their performance degenerates. Feature selection for SVM has been widely studied. For example, [WMC+01] introduced an algorithm based upon finding the features which minimize bounds on the leave-one-out error. The search can be efficiently performed via gradient descent. [FCS03] proposed an approach that takes existing theoretical bounds on the generalization error for SVMs instead of performing cross-validation. This is computationally faster than k-fold cross-validation. Additionally, in general, the error bounds have a higher bias than cross-validation in practical situations they often have a lower variance and can thus reduce the overfitting of the wrapper algorithm. A convex energy-based framework to jointly perform feature selection and SVM parameter learning for linear and non-linear kernels was proposed by [NDlT10] . They also showed the equivalence between their approach and the ℓ_1 SVM. In a recent work, [LEP19] developed an efficient method for sparse support vector machines with ℓ_0 norm approximation. The proposed method approximates the ℓ_0 minimization through solving a series of ℓ_2 optimization problems, which can be formulated with dual variables

Furthermore, in practical applications, training samples are often contaminated by noise and some even have wrong labels [FV13]. These are usually known as outliers. In order to mitigate the effects of outliers, different approaches have been proposed to improve the robustness of SVM. [SHX02] suggested to use the distance between each training sample and its class center to calculate an adaptive margin so as to reduce the influence of outliers. Weighted SVM (WSVM) or fuzzy SVM was also proposed to deal with outliers [WL13, LW02, BP10]. In WSVM, different weights are assigned to different training samples which can show their importance in the training data set. Several weight functions have been proposed [WL13, LW02, BP10]. [DX15] presented a novel combinatorial technique, which was called random gradient descent (RGD) tree, to identify and remove outliers in SVM and developed a new algorithm called RGD-SVM. [XCHP17] proposed the re-scaled hinge loss which is a monotonic, bounded and non-convex loss. Introducing a Ramp Loss function into one-class SVM optimization to reduce outliers influence was

suggested by [XWX17]. Then the outliers are identified and removed from the training set. The final classification surface is obtained on the remaining training samples. [YD19] introduced a new robust loss function (called L_q loss) based on the concept of quantile and correntropy, which can be seen as an improved version of quantile loss function. To deal with label outliers, [BDPZ18] introduced a variable $\Delta y_i \in \{0,1\}$ where 1 indicates that the label was incorrect and has in fact been flipped, and 0 indicates that the label was correct.They also introduced a variable Δx_i to deal with uncertainty of features. They proposed the use of mixed integer optimization problems to solve the obtained problem. However, the algorithm is not sparse.
To obtain a sparse and robust least squares support vector machines (SR-LSSVM), [CZ18] proposed the SR-LSSVM algorithm to obtain a sparse solution of the robust least squares SVM (R-LSSVM) [WZ14, YTH14] by applying a low-rank approximation of the kernel matrix.

Contributions:

In this chapter, we address the problem of both feature selection and outlier detection using the ℓ_0 norm. We summarize our contributions in this paper below:

— We present an approach jointly performing feature selection and outlier detection for SVM classification;

— We propose to recast the presented problem as a mixed integer optimization problem which allows the use of efficient solvers (Gurobi) to solve it. Note that the suboptimality of the obtained solution is guaranteed even if we terminate the algorithm early;

— We present computational results on both real and synthetic datasets and compare the proposed approach with the classical 0-1 loss and hinge loss classification problems. The results show that the proposed approach provides high quality solutions.

5.3.1 Linear Binary Classification

We have n training points, where each input x_i has p attributes and is in one of two classes $y_i \in \{-1,1\}$. Under linear assumption, the classification function can be expressed as $f(x,w) = w^{\mathrm{T}}x + b$. The goal is to predict the target class $\hat{y} \in \{-1,1\}$ which is defined by:

$$\hat{y}_i = \begin{cases} 1 & f(x_i,w) \geq 0 \\ -1 & f(x_i,w) < 0\,. \end{cases} \tag{5.3}$$

The natural way to quantify the performance of a classifier is using the 0-1 loss function: for a given instance x and a true binary label $y \in \{-1,1\}$, we incur a loss of $1 if\ sign(yf) < 0$, and 0 otherwise, that is:

$$\mathbb{1}_{\{y \neq sign(f(x,w))\}} = \begin{cases} 1 & \text{if } y \neq sign(f(x,w)) \\ 0 & otherwise\,, \end{cases} \tag{5.4}$$

where $\mathbb{1}_A$ denotes the indicator function of set A.

The 0-1 loss classification problem can be written as

$$\min \sum_{i=1}^{n} \mathbb{1}_{\{y_i \neq sign(f(x_i,w))\}}\,. \tag{5.5}$$

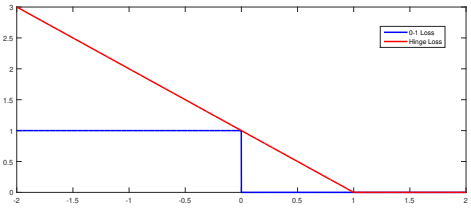

FIGURE 5.1 – Illustration of the hinge loss which is a convex surrogate to the 0- loss. The 0-1 loss is shown in blue and the hinge loss is shown in red.

Problem (5.5) is non-convex, to this end it has been replaced by a convex surrogate such as the hinge loss. However, advances in integer optimization resulted an impressive speedup in solving mixed integer optimization problems (MIO). To this end, [TLX+14] proposed to recast the problem of 0-1 loss classification (5.5) as a mixed integer optimization problem, that is:

$$\begin{array}{ll} \min & \sum_{i=1}^{n} l_i \\ \text{s.t.} & y_i(w^{\mathrm{T}} x_i + b) \geq 1 - \mathrm{M} l_i \\ & l \in \{0,1\}^n. \end{array} \tag{5.6}$$

where M is a sufficiently large constant. Since this formulation suffers from infinite number of optimal solutions and it lacks from the generalization ability, [TLX+14] proposed a maximum margin 0-1 loss classifier defined as follows:

$$\begin{array}{ll} \min & \sum_{i=1}^{n} l_i + \mathrm{C} w^{\mathrm{T}} w \\ \text{s.t.} & y_i(w^{\mathrm{T}} x_i + b) \geq 1 - \mathrm{M} l_i \\ & l \in \{0,1\}^n \end{array} \tag{5.7}$$

where C is a positive parameter, and showed the efficiency of this approach for small-scale classification problems.

Introducing Binary Variables

Variable selection involves the ℓ_0 norm function to count the number of useful variables. This counting function can be represented by introducing p binary variables $z_j \in \{0,1\}$ such that

$$\|w\|_0 = \sum_{j=1}^{p} z_j \quad \text{and} \quad z_j = 0 \Leftrightarrow w_j = 0.$$

Different approaches can be used to force $z_j = 0 \Leftrightarrow w_j = 0$ into an optimization problem, such as:

1. Replace w_j by $z_j w_j$ for $j = 1, \ldots, p$,
2. Set $|w_j|(1 - z_j) = 0$ for $j = 1, \ldots, p$ or $\sum_{j=1}^{p} |w_j|(1 - z_j) = 0$,
3. Use a big-M constraint, $|w_j| \leq \mathrm{M}_v z_j$ for $j = 1, \ldots, p$ and for some fixed constant M_v large enough (such as $\mathrm{M}_v \geq \max_j |w_j^\star|$, $w_j^\star$ being the solution of the optimization problem),
4. Treat $z_j = 0 \Leftrightarrow w_j = 0$ as logical implications (also called indicator constraints or special ordered set SOS-1). Note that this kind of logical implication can be efficiently handled in a branch-and-bound procedure for MIO problems.

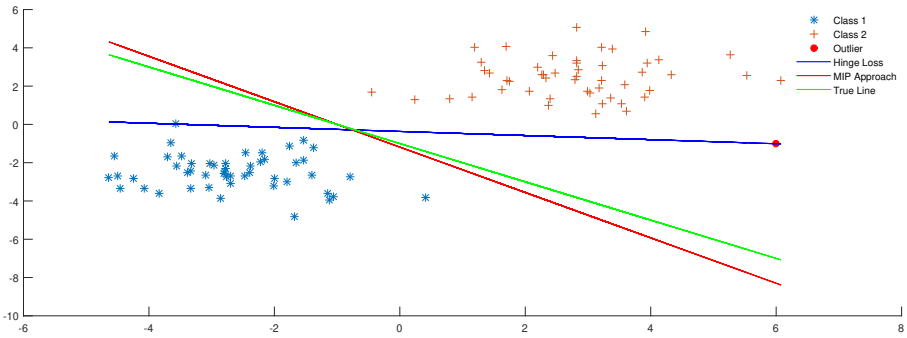

FIGURE 5.2 – Example of Synthetically Generated Data in Two Dimensions to Show the Effect of an Outlier on the Hinge-Loss Classification. The True Generating Hyperplane in Green, the Hinge-loss Hyperplane in Blue and the MIP Approach Hyperplane in Red.

We now discuss and give a short overview of the advantages and drawbacks of each approach. The two first approaches involve nonlinear interaction terms between binary and continuous variables. Their interest lies in the possibility of obtaining interesting continuous relaxations. The main advantage of the big M method (approach 3) is that it brings only linear inequality constraints but the value of the M term needs to be chosen carefully since it shows a great deal of practical influence on the solver performance. Logical implications (approach 4) have the advantage of avoiding these types of problems, as they do not rely on a separate constant value. However, they tend to have weaker relaxations, a condition which may lead to longer solve times in a model. In this paper we will use the third approach for our implementation.

Our Approach

To deal with the problem of outlier detection, we propose to add a variable τ so that Problem (5.2) becomes:

$$\begin{array}{lll} \min\limits_{w,\xi,\tau} & \frac{1}{2}\|w\|_2^2 + C\sum\limits_{i=1}^{n}|\xi_i - \tau_i| & \\ \text{s.t.} & y_i(w^T x + b) \geq 1 - \xi_i & i = 1\dots n \\ & ||w||_0 \leq k_v & \\ & ||\tau||_0 \leq k_o & \\ & \xi_i \geq 0 & i = 1\dots n. \end{array} \tag{5.8}$$

where the ℓ_0 norm of a vector w counts the number of nonzeros in w. We note that in Problem (5.8), $\tau(i) \neq 0$ means that the observation "i" is an outlier.

A MIP Formulation

To solve (5.8) exactly, we recast it as a mixed integer optimization problem. Two binary variables z and t are introduced to control the sparsity levels for w and τ respectively. The

MIP formulation of (5.8) is as follows:

$$\begin{array}{lll} \min\limits_{w,\xi,\tau,t,z,b} & \frac{1}{2}\|w\|_2^2 + C\sum\limits_{i=1}^{n}|\xi_i - \tau_i| & \\ \text{s.t.} & \sum\limits_{j=1}^{p} z_j \le k_v & \\ & |w_j| \le z_j M_v & j = 1\dots p \\ & \sum\limits_{i=1}^{n} t_i \le k_o & \\ & |\tau_i| \le t_i M_o & i = 1\dots n \\ & y_i(w'x_i + b) \ge 1 - \xi_i & i = 1\dots n \\ & \xi_i \ge 0 & i = 1\dots n. \end{array} \tag{5.9}$$

where $w \in \mathbb{R}^p$, $\tau, \xi \in \mathbb{R}^n$, $t \in \{0,1\}^n$, $z \in \{0,1\}^p$ and $b \in \mathbb{R}$.

When $k_v = 0$ and $k_o = 0$, no feature selection nor outlier detection are performed, the resulting problem is the classical hinge loss classification problem. In the above formula, M_v and M_o are two big values.

Solving the Problem Using Gurobi

To overcome the absolute value in the objective function, we introduce two new variables α^+ and α^-, such that $\xi_i - \tau_i = \alpha_i^+ - \alpha_i^-$, and $|\xi_i - \tau_i| = \alpha_i^+ + \alpha_i^-$, where $\alpha_i^+, \alpha_i^- \ge 0$ for $i = 1\dots n$. Then the new obtained problem is as follows:

$$\begin{array}{lll} \min\limits_{w,\xi,\tau,t,z,b} & \frac{1}{2}\|w\|_2^2 + C\sum\limits_{i=1}^{n}(\alpha_i^+ + \alpha_i^-) & \\ \text{s.t.} & \sum\limits_{j=1}^{p} z_j \le k_v & \\ & |w_j| \le z_j M_v & j = 1\dots p \\ & \sum\limits_{i=1}^{n} t_i \le k_o & \\ & |\tau_i| \le t_i M_o & i = 1\dots n \\ & y_i(w'x_i + b) \ge 1 - \xi_i & i = 1,\dots,n \\ & \xi_i - \tau_i = \alpha_i^+ - \alpha_i^- & i = 1\dots n \\ & \xi_i \ge 0 & i = 1\dots n \\ & \alpha_i^+ \ge 0 & i = 1\dots n \\ & \alpha_i^- \ge 0 & i = 1\dots n, \end{array} \tag{5.10}$$

where $w \in \mathbb{R}^p$, $\tau, \xi, \alpha^+, \alpha^- \in \mathbb{R}^n$, $t \in \{0,1\}^n$, $z \in \{0,1\}^p$ and $b \in \mathbb{R}$.

Computational Cost

In Figure 5.3, the left panel shows the evolution of upper and lower bounds with time when $k_v = p$, while the right panel shows this evolution when $k_v = 0.8p$. By comparing the left and the right panels, we can see that the computational time increased from 1200 seconds to 1800 seconds (top panel) and from 4 seconds to 8 seconds (bottom panel). This means that the value of k_v has an influence on the computational cost.
Similarly, the top panel shows the evolution of upper and lower bounds with time when $k_o = 5\%$, while the bottom panel shows this evolution when $k_o = 2.5\%$. A simple comparison between the top and the bottom panels sheds the light on how much increasing

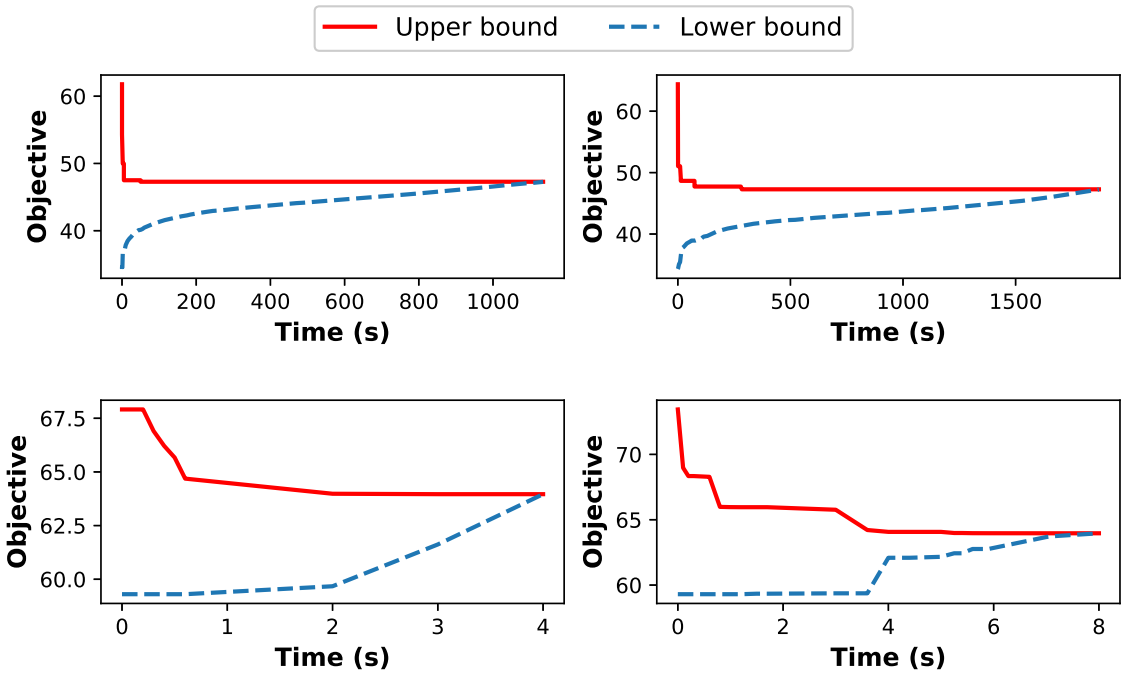

FIGURE 5.3 – The evolution of the MIO for the breast cancer prognostic data set with $n = 194$ and $p = 33$. The top panel shows the evolution of upper and lower bounds with time when $k_o = 5\%$, while the bottom panel shows the evolution of upper and lower bounds with time when $k_o = 2.5\%$. The left panel shows the evolution of upper and lower bounds with time when $k_v = p$, while the right panel shows the evolution of upper and lower bounds with time when $k_v = 0.8p$. For all panels, C = 1.

the value of k_o (percentage of outliers to detect) will increase the time needed to certify optimality. Indeed, decreasing k_o from 5% to 2.5% resulted a significant decrease of the computational cost, that is from 1200 seconds to only 4 seconds, and from 1800 seconds to only 8 seconds.
We note that optimal solutions are found in a few seconds in the top panel examples, but it takes 20-30 minutes to certify optimality via the lower bounds. We also note that the computational time depends on the value of C and the big-M values.

5.3.2 Experiments on synthetic data sets

To report the robustness of the proposed approach, we evaluated its performance on synthetically generated data sets. In these experiments, we run the classical hinge-loss classifier and the MIP approach to recover the separating hyperplane classifier.

Experimental setup

The experiment uses data in $\mathbb{R}^2$. The data are generated synthetically as follows:

1. Twenty-five points are generated as multivariate random normal, $N(3.5e, I)$ where e is the vector of ones and I is the identity matrix. These points are given the label +1.
2. Twenty-five points are generated as multivariate random normal, $N(-3.5e, I)$. These points are given the label −1.
3. Ten outlier points are introduced as multivariate random normal $N(0, 3I)$, where 0 is the vector of zeros. The labels are randomly generated as either -1 or +1.

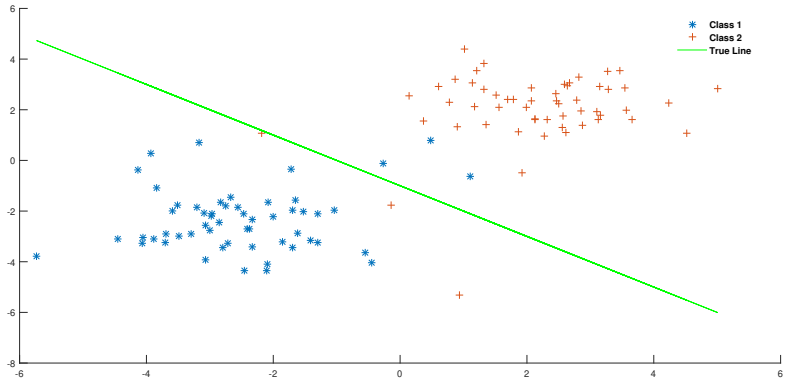

FIGURE 5.4 – Example of Synthetically Generated Data in Two Dimensions Alongside the True Generating Hyperplane.

TABLE 5.1 – Performance results for synthetic data experiments.

	Accuracy	Similarity
Hinge Loss	96.93	0.9428
MIP Approach	**97.85**	**0.9813**

We split the data 75% / 25% into training and validation sets, which we used to tune the parameters for both methods. To create the test set, we generated 1000 points in the same way as items 1 and 2 above.

An example of a data set generated according to this procedure is shown in Figure 5.4. By the symmetry of this data generation process, we can see that the true hyperplane separating the two clusters of points is given by $e^T x = 0$. The goal of the experiment is to show how closely the two methods can recover the truth in the data. We are interested in the following two measures:

— Accuracy: We measure and evaluate the out-of sample accuracy of the trained classifiers on the test set, defined by:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN},$$

where TP and TN represent the quantity of correct positive and correct negative samples, respectively; FN and FP respectively represent the number of misclassification negative and positive samples. The higher the values of the Accuracy, the better the model is.

— Similarity: To evaluate the ability of each method to recover the truth in the data, we measure the cosine of the angle between the separating hyperplane generated by the methods and the true hyperplane.

We recall that the cosine of the angle α between two vectors u and v is given by:

$$cos(\alpha) = \frac{u.v}{||u|| \times ||v||}.$$

Results

This experiment was repeated 1000 times. We present the means of the two measures for each method in Table 5.1. The results show that the MIP approach improved the per-

formance of classification. In fact, the accuracy increased by about 1% and that it recovered the truth better than the classical hinge loss classifier (cosine value closer to 1 means smaller angle between hyperplanes and thus better recovery).

5.3.3 Experiments on real data sets

To evaluate the effectiveness of the proposed method, we carry out numerical simulations on twelve real-world data sets. All experiments are implemented using MATLAB-Gurobi interface. The experiment environment is: PC with Intel Core i7 4700MQ (2.40 GHz) with 8 GB memory. We note that for each problem instance, we used a time limit of 15 minutes for Gurobi to optimize the classification problem.
We recall that to obtain: the hinge-loss classification problem solution we solved Problem (5.2), the 0-1 loss problem solution we solved Problem (5.7). The solution of the MIP approach was found by solving Problem (5.8).

Experimental Setup

To evaluate the performance of the proposed approach, we considered two scenarios:

1. In the first scenario, 10% of the training and validation sets labels were randomly flipped. The aim is to study the robustness of the mixed integer programming approach.
2. In the second scenario, we wanted to mimic real-world setting, hence data sets were not modified.

For both scenarios, each data set was normalized using the min-max scaling and was split into three parts: the training set (60%), the validation set (20%), and the testing set (20%). The training set was used to train each classifier for a variety of combinations of input parameters. For each combination of parameters, the accuracy on the validation set was calculated, and this was used to select the best combination of parameters for each classifier. Finally, the classifier was trained by using these best parameters on the combined training and validation sets, before reporting the out-of-sample accuracy on the testing set. All methods were trained, validated, and tested on the same random splits, and computational experiments were repeated five times for each data set with different splits. For each data set and classification method, we report the average out-of-sample accuracy across all five splits. C was chosen from the set $[10^{-4}, 10^{-3}, \ldots, 10^{4}]$, k_v was set to $k_v = p$ for the first scenario, and chosen from the set $[p, 0.8p, 0.6p]$ for the second scenario that is no feature selection was performed, 80% and 60% of features are selected respectively. k_o was chosen from the set $[0.025n, 0.05n, 0.1n]$ that is 2.5%, 5% and 10% of outliers to be detected respectively.

5.3.4 Results

Tables 5.2 and 5.3 present the means of the accuracy for each method. The robustness of the proposed approach is shown in Table 5.2. In fact, it had a superior performance on 9 data sets, and a tie for one data set, when 10% of labels were flipped. An important remark is that no variable selection was performed during this scenario so the comparison between the MIP approach and the hinge-loss classification is based only on the robustness of the MIP approach. This side by side comparison sheds the light on the significant improvement obtained with the MIP approach.
The second scenario is closer to the real world setting. The data sets are taken without

TABLE 5.2 – Out of sample accuracy averaged across five seeds for each classification method on all data sets. (first scenario).

	n	p	Hinge loss	0-1 loss	MIP approach
Arrythmia	68	280	52.31	**64.62**	**64.62**
Breast Cancer Coimbra	116	9	65.22	60.87	**72.17**
Breast Cancer Prognostic	194	33	63.16	78.42	**84.74**
Connections Bench Sonar	208	60	65.17	72.20	**75.61**
Fertility	100	9	64.00	78.00	**86.00**
Ionosphere	351	33	63.71	84.86	**85.43**
Monks-1	124	6	65.83	**72.50**	71.67
Monks-2	169	6	**65.45**	63.03	60.51
Monks-3	122	6	82.50	77.21	**83.33**
Pima	768	8	56.60	68.37	**76.73**
Spect Heart	80	22	65.00	72.50	**75.92**
Spectf Heart	80	44	78.75	79.25	**81.25**

TABLE 5.3 – Out of sample accuracy averaged across five seeds for each classification method on all data sets. (second scenario).

	n	p	Hinge loss	0-1 loss	MIP approach
Arrythmia	68	280	70.76	69.23	**81.53**
Breast Cancer Coimbra	116	9	**73.04**	70.43	70.43
Breast Cancer Prognostic	194	33	76.84	78.94	**81.05**
Connections Bench Sonar	208	60	72.19	**76.58**	**76.58**
Fertility	100	9	86.00	86.00	**88.00**
Ionosphere	351	33	84.28	82.57	**85.14**
Monks-1	124	6	62.50	**67.51**	64.98
Monks-2	169	6	**61.21**	59.79	**61.21**
Monks-3	122	6	79.16	82.50	**82.78**
Pima	768	8	78.30	78.21	**78.82**
Spect Heart	80	22	63.75	67.50	**70.83**
Spectf Heart	80	44	70.00	71.25	**77.50**

TABLE 5.4 – Improvement Due to Robustness of MIP Approach, Comparing the Hinge-Loss Classifier to the Robust MIP Approach (second scenario).

Name of the dataset	*Percentage of MIP Robust Improvement*
Arrythmia	10.77 %
Breast Cancer Coimbra	- 2.61 %
Breast Cancer Prognostic	4.21 %
Connections Bench Sonar	4.39 %
Fertility	2.00 %
Ionosphere	0.86 %
Monks-1	2.48 %
Monks-2	0.00 %
Monks-3	3.62 %
Pima	0.52 %
Spect Heart	7.08 %
Spectf Heart	7.50 %
Mean	**3.40 %**

any change or modification. From Table 5.3, it is clear that the prediction accuracy of our approach is higher than those of the compared algorithms for almost all datasets. We can remark a significant accuracy improvement for some datasets. For example, we obtained about 11% improvement for Arrythmia dataset. In general, it can be seen that the proposed approach provides high quality solutions. We also note that the pairwise comparison of the 0-1 classification against the hinge loss classification shows that none of the two losses dominates the other. Indeed each loss showed better results on six data sets, while a tie was obtained for one data set. An important caveat to emphasize upfront is that the ℓ_0 robust regression algorithm was given 15 minutes time limit per problem instance per subset size. This practical restriction may have caused this algorithm to under perform in some cases.

Table (5.4) shows the improvement in out-of-sample accuracy of the MIP approach over the classical hinge-loss classification in-sample accuracy in the second scenario. We defined the robust improvement as the difference in out-of-sample accuracy between the methods. We note that over 11 from 12 data sets, the MIP approach over performed the classical hinge loss classifier.

5.4 Conclusion

In this chapter, we propose a method for support vector machine which solves the underlying optimization problem that handles both feature selection and outlier detection. We formulate the problem as a mixed integer optimization problem and use an efficient commercial solver (Gurobi) to solve it. Furthermore, we present an empirical comparison between this method, the classical hinge-loss and the 0-1 loss classification methods. The experimental results have verified the superior performance of the proposed method. In terms of computational efficiency, the MIP solution can already be adopted for relatively small data sets. For the high dimensional case, a screening procedure would be suggested to reduce the computational cost.

Chapitre 6

Conclusion and Future Work

Sommaire

6.1 Conclusion

This monograph focused on the optimization problems of robust and sparse linear regression and support vector machines. The problems consists of using double ℓ_0 norm constraints where the ℓ_0 norm can explicitly count the number of non-zero coefficients, nevertheless with the characteristics of non-smoothness and non-convexity, which makes the problems NP-hard. To obtain the optimal solution, each problem was formulated using mixed integer programming techniques.

In Chapter 2, we make all the preliminary definitions needed in the context of the thesis were surveyed . A mathematical background on optimization was introduced. Furthermore, convex and non-convex optimization algorithms were discussed briefly together with an introduction to linear and mixed integer programming.

In Chapter 3 we proposed a robust adaption of best subset selection that is highly resistant to contamination in both the response and the predictors. Our estimator generalized the notion of subset selection to both predictors and observations, thereby achieving robustness in addition to sparsity. This procedure is defined by a combinatorial optimization problem for which we apply modern discrete optimization methods. We formulated the problem as a mixed integer programming to find a provably optimal solution for the proposed problem.

Our approach has the appealing characteristic that if we terminate the algorithm early, we obtain a solution with a guarantee on its suboptimality. Furthermore, our framework can accommodate side constraints on β. We also introduced a general algorithmic framework based on a discrete extension of first order continuous optimization methods that provide near-optimal solutions for the best subset problem. The MIP algorithm significantly benefits from solutions obtained by the first order methods and problem specific information that can be computed in a data-driven fashion. An empirical comparison of the proposed method, its ℓ_1 relaxation, the best subset selection problem, lasso and PALM was introduced on real and synthetic data sets. Experimental results showed the high quality solutions obtained by the MIP algorithm. However, the ℓ_1 relaxation performed better

for low SNR values due to the overfitting behavior of the ℓ_0 norm.

In Chapter 4, we study the overfitting aspect of the robust and sparse linear regression model causing its under-perfomance for low SNR values. While the ℓ_0 norm procedure is often perceived as the "gold standard" in sparse learning when the signal to noise ratio (SNR) is high, its predictive performance deteriorates when the SNR is low. In particular, it is outperformed by continuous shrinkage methods, such as the Lasso. We investigate the behavior double ℓ_0 norm constraints in the low SNR regimes and propose an alternative approach based on a regularized version of the ℓ_0 criterion. Our proposed estimators (a) mitigated, to a large extent, the poor predictive performance of best-subset selection in the low-SNR regimes; and (b) performed favorably, while generally delivering substantially sparser models, relative to the best predictive models available via the Lasso. We conducted an extensive theoretical analysis of the predictive properties of the proposed approach and provided justification for its superior predictive performance relative to best-subsets selection when the SNR is low. Our estimators can be expressed as solutions to mixed integer second order conic optimization problems and, hence, are amenable to modern computational tools from mathematical optimization. Furthermore, a discrete first order optimization algorithm was also proposed to solve the obtained problem. Experimental results on both real and synthetic data sets showed that the added ℓ_1 penalty term improved the performance of the ℓ_0 norm problem for low SNR values. In addition, the discrete first order algorithm performed very well and it was suggested to use it especially for the high dimensional case.

In Chapter 5, motivated by the fact that there may be inaccuracies in the training data, we applied robust optimization techniques to study in a principled way the uncertainty in data in classification problems and obtained robust formulations for the one of the most widely used classification methods: support vector machines. In addition, to control the level of sparsity, we added a cardinality constraint on the weights vector. We demonstrated the advantage of this robust and sparse formulation over two classical SVM methods: the 0-1 loss and hinge loss classification problems. We run large-scale computational experiments across a sample of 12 data sets from the University of California Irvine Machine Learning Repository and showed that the proposed method improved the accuracy in the majority of the data sets. We observed significant gains for the robust and sparse classification method when outliers were added to data sets and when data sets were used without any modification.

6.2 Future Work

The study of the exact ℓ_0 norm optimization problem for variable selection and outlier detection is a promising research topic in statistical learning. The optimal solution can be obtained by formulating the problem as a mixed integer programming and solving it using an efficient solver. However, these problems are $\mathcal{NP}$-hard and they are not scalable to be used for high dimensional data. A possible way to address this scalability issue would be to adapt heuristic and metaheuristic algorithms for combinatorial optimization.

For the linear regression model, we proposed a proximal alternating minimization algorithm that provides a near-optimal solutions for the proposed problems. We also showed the high quality of the solutions provided by this algorithm even though these solutions are not optimal. To solve the problems exactly in a short time, we suggest to use a dedicated branch and bound methodology as in [MBM$^+$20]. Such approach will ensure

solving the MIP problems exactly more efficiently than the optimization solvers, which makes the NP-hard problems more scalable to be used for high dimensional regimes.

For the robust and sparse support vector machines problem we propose the transformation of the original optimization problem by using the Lagrangian multipliers. Such transformation would decrease the computational time of providing a solution. As already pointed out, the shortcoming of the exact ℓ_0 norm method is its high computational complexity. Fortunately, we can benefit from the fact that the optimization theory is well developing and the computational capability of the machine is highly improving.

References

[AADFG20] Umberto Amato, Anestis Antoniadis, Italia De Feis, and Irene Gijbels. Penalised robust estimators for sparse and high-dimensional linear models. *Statistical Methods & Applications*, pages 1–48, 2020.

[AB94] David W Aha and Richard L Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the AAAI-94 workshop on Case-Based Reasoning*, volume 106, page 112, 1994.

[ACG+13] Andreas Alfons, Christophe Croux, Sarah Gelper, et al. Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *The Annals of Applied Statistics*, pages 226–248, 2013.

[AGN11] Anestis Antoniadis, Irène Gijbels, and Mila Nikolova. Penalized likelihood regression for generalized linear models with non-quadratic penalties. *Annals of the Institute of Statistical Mathematics*, pages 585–615, 2011.

[Aus71] A Auslender. Méthodes numériques pour la décomposition et la minimisation de fonctions non différentiables. *Numerische Mathematik*, pages 213–223, 1971.

[BBE+03] Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, pages 1229–1243, 2003.

[BDPZ18] Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, and Ying Daisy Zhuo. Robust classification. *INFORMS Journal on Optimization*, pages 2–34, 2018.

[BDV11] Stephen Boyd, John Duchi, and Lieven Vandenberghe. Subgradients, lecture notes for EE364b, 2011.

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Bix12] Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121, 2012.

[BJK15] Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems*, pages 721–729, 2015.

[BJQS14] Chenglong Bao, Hui Ji, Yuhui Quan, and Zuowei Shen. L0 norm based dictionary learning by proximal methods with global convergence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3858–3865, 2014.

[BKM67] EML Beale, MG Kendall, and DW Mann. The discarding of variables in multivariate analysis. *Biometrika*, pages 357–366, 1967.

[BKM15] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *Ann. Statist.*, pages 2324–2354, 2015.

[BKM16] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The annals of statistics*, pages 813–852, 2016.

[BP10] Rukshan Batuwita and Vasile Palade. Fsvm-cil: fuzzy support vector machines for class imbalance learning. *IEEE Transactions on Fuzzy Systems*, pages 558–571, 2010.

[BRF11] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In *Advances in neural information processing systems*, pages 2115–2123, 2011.

[BRT+09] Peter J Bickel, Ya'acov Ritov, Alexandre B Tsybakov, et al. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, pages 1705–1732, 2009.

[BST14] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, pages 459–494, 2014.

[BT03] Dimitri P Bertsekas and John N Tsitsiklis. Parallel and distributed computation: numerical methods. 2003.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[BVDG11] Peter Bühlmann and Sara Van De Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[BW05] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005.

[CCM13] Yudong Chen, Constantine Caramanis, and Shie Mannor. Robust sparse regression under adversarial corruption. In *International Conference on Machine Learning*, pages 774–782, 2013.

[CFZ09] Bertrand Clarke, Ernest Fokoue, and Hao Helen Zhang. *Principles and theory for data mining and machine learning*. Springer Science & Business Media, 2009.

[Coo77] R Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, pages 15–18, 1977.

[CP+09] Emmanuel J Candès, Yaniv Plan, et al. Near-ideal model selection by l1 minimization. *The Annals of Statistics*, pages 2145–2177, 2009.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, pages 273–297, 1995.

[CZ18] Li Chen and Shuisheng Zhou. Sparse algorithm for robust lssvm in primal space. *Neurocomputing*, pages 2880–2891, 2018.

[DBJ+20] Pascaline Descloux, Claire Boyer, Julie Josse, Aude Sportisse, and Sylvain Sardy. Robust lasso-zero for sparse corruption and model selection with missing covariates. *arXiv preprint arXiv:2005.05628*, 2020.

[Des19] Pascaline Descloux. *Sparse Support Recovery with Thresholded Basis Pursuit and Lasso-Zero, and an Extension to Handle Missing Data*. PhD thesis, University of Geneva, 2019.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, pages 1–22, 1977.

[DP04] Kalyanmoy Deb and Koushik Pal. Efficiently solving: A large-scale integer linear program using a customized genetic algorithm. In *Genetic and Evolutionary Computation Conference*, pages 1054–1065. Springer, 2004.

[DT19] Arnak Dalalyan and Philip Thompson. Outlier-robust estimation of a sparse linear model using l-1penalized huber's m-estimator. In *Advances in Neural Information Processing Systems*, pages 13188–13198, 2019.

[DX15] Hu Ding and Jinhui Xu. Random gradient descent tree: A combinatorial approach for svm with outliers. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[EHJ+04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, pages 407–499, 2004.

[FCS03] Holger Frohlich, Olivier Chapelle, and Bernhard Scholkopf. Feature selection for support vector machines by means of genetic algorithm. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 142–148. IEEE, 2003.

[FHST16] J Friedman, T Hastie, N Simon, and R Tibshirani. Lasso and elastic-net regularized generalized linear models. r-package version 2.0-5. 2016, 2016.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[FHT10] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, page 1, 2010.

[Fre09] David A Freedman. *Statistical models: theory and practice*. cambridge university press, 2009.

[Fri91] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[FV13] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, pages 845–869, 2013.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, pages 1157–1182, 2003.

[GK09] Rahul Garg and Rohit Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 337–344, 2009.

[GP02] A Giloni and M Padberg. Least trimmed squares regression, least median squares regression, and mathematical programming. *Mathematical and Computer Modelling*, pages 1043–1060, 2002.

[Gru69] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, pages 1–21, 1969.

[HA04] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, pages 85–126, 2004.

[Heb49] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[HK70] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, pages 55–67, 1970.

[HL67] RR Hocking and RN Leslie. Selection of the best subset in regression analysis. *Technometrics*, pages 531–540, 1967.

[HRRS11] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011.

[HTT] Trevor Hastie, Robert Tibshirani, and Ryan J Tibshirani. Best subset, forward stepwise, or lasso?

[iG04] J Tropp Greed is Good. Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, pages 2231–2242, 2004.

[LD10] Ailsa H Land and Alison G Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer, 2010.

[LDB09] Jason N Laska, Mark A Davenport, and Richard G Baraniuk. Exact signal recovery from sparsely corrupted measurements through the pursuit of justice. In *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, pages 1556–1560. IEEE, 2009.

[Lee10] Yoonkyung Lee. Support vector machines for classification: a statistical portrait. In *Statistical Methods in Molecular Biology*, pages 347–368. Springer, 2010.

[LEP19] Zhenqiu Liu, David Elashoff, and Steven Piantadosi. Sparse support vector machines with l0 approximation for ultra-high dimensional omics data. *Artificial intelligence in medicine*, pages 134–141, 2019.

[Li13] Xiaodong Li. Compressed sensing and matrix completion with constant proportion of corruptions. *Constructive Approximation*, pages 73–99, 2013.

[Lin17] Jeff Linderoth. Mixed-integer (linear) programming: Recent advances, and future research directions. In *Proceedings of FOCAPO*, 2017.

[LJK+00] Jorma Laurikkala, Martti Juhola, Erna Kentala, N Lavrac, S Miksch, and B Kavsek. Informal identification of outliers in medical data. In *Fifth international workshop on intelligent data analysis in medicine and pharmacology*, volume 1, pages 20–24, 2000.

[LL11] Jon Lee and Sven Leyffer. *Mixed integer nonlinear programming*, volume 154. Springer Science & Business Media, 2011.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, pages 129–137, 1982.

[LM07] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection*. CRC Press, 2007.

[LM15] Johannes Lederer and Christian Müller. Don't fall for tuning parameters: tuning-free variable selection in high dimensions with the trex. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[LPZ11] Gaorong Li, Heng Peng, and Lixing Zhu. Nonconcave penalized m-estimation with a diverging number of parameters. *Statistica Sinica*, pages 391–419, 2011.

[LW02] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *IEEE transactions on neural networks*, pages 464–471, 2002.

[LZ08] Youjuan Li and Ji Zhu. L 1-norm quantile regression. *Journal of Computational and Graphical Statistics*, pages 163–185, 2008.

[Mal75] CL Mallows. On some topics in robustness. unpublished memorandum, bell tel. *Laboratories, Murray Hill*, 1975.

[MBM+20] R. B. Mhenni, S. Bourguignon, M. Mongeau, J. Ninin, and H. Carfantan. Sparse branch and bound for exact optimization of l0-norm penalized least squares. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5735–5739, 2020.

[Mil02] Alan Miller. *Subset selection in regression*. CRC Press, 2002.

[MLG+18] Yan Ma, Kang Liu, Zhibin Guan, Xinkai Xu, Xu Qian, and Hong Bao. Background augmentation generative adversarial networks (bagans): Effective data generation based on gan-augmented 3d synthesizing. *Symmetry*, page 734, 2018.

[MRD17] Rahul Mazumder, Peter Radchenko, and Antoine Dedieu. Subset selection with shrinkage: Sparse linear modeling when the snr is low. *arXiv preprint arXiv:1708.03288*, 2017.

[MT15] Ryuhei Miyashiro and Yuichi Takano. Subset selection by mallows' cp: A mixed integer programming approach. *Expert Systems with Applications*, pages 325–331, 2015.

[Nat95] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, pages 227–234, 1995.

[NDlT10] Minh Hoai Nguyen and Fernando De la Torre. Optimal feature selection for support vector machines. *Pattern recognition*, pages 584–591, 2010.

[Nes13] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[NT12] Nam H Nguyen and Trac D Tran. Robust lasso with missing and grossly corrupted observations. *IEEE transactions on information theory*, pages 2036–2058, 2012.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[ÖAC16] Viktoria Öllerer, Andreas Alfons, and Christophe Croux. The shooting s-estimator for robust regression. *Computational Statistics*, pages 829–844, 2016.

[Pow73] Michael JD Powell. On search directions for minimization algorithms. *Mathematical programming*, pages 193–201, 1973.

[RH18] Peter J Rousseeuw and Mia Hubert. Anomaly detection by robust statistics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1236, 2018.

[RL05] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.

[Rou84] Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, pages 871–880, 1984.

[RPD01] John O Rawlings, Sastry G Pantula, and David A Dickey. *Applied regression analysis: a research tool*. Springer Science & Business Media, 2001.

[RY84] Peter Rousseeuw and Victor Yohai. Robust regression by means of s-estimators. In *Robust and nonlinear time series analysis*, pages 256–272. Springer, 1984.

[SBC+17] Weijie Su, Małgorzata Bogdan, Emmanuel Candes, et al. False discoveries occur early on the lasso path. *The Annals of statistics*, pages 2133–2150, 2017.

[SDDO03] Hervé Stoppiglia, Gérard Dreyfus, Rémi Dubois, and Yacine Oussar. Ranking a random feature for variable and feature selection. *Journal of machine learning research*, (Mar):1399–1414, 2003.

[SHX02] Qing Song, Wenjie Hu, and Wenfang Xie. Robust support vector machine with bullet hole image classification. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, pages 440–448, 2002.

[Ska94] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning Proceedings 1994*, pages 293–301. Elsevier, 1994.

[SNW12] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.

[SO11] Yiyuan She and Art B Owen. Outlier detection using nonconvex penalized regression. *Journal of the American Statistical Association*, pages 626–639, 2011.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, pages 267–288, 1996.

[Tib11] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 273–282, 2011.

[TLX+14] Yufang Tang, Xueming Li, Yan Xu, Shuchang Liu, and Shuxin Ouyang. A mixed integer programming approach to maximum margin 0–1 loss classification. In *2014 International Radar Conference*, pages 1–6. IEEE, 2014.

[Tse01] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, pages 475–494, 2001.

[VdG+08] Sara A Van de Geer et al. High-dimensional generalized linear models and the lasso. *The Annals of Statistics*, pages 614–645, 2008.

[Vie15] Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, pages 3–57, 2015.

[Wei05] Sanford Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

[Wik20] Wikipedia. Linear programming — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Linear\%20programming&oldid=978990729`, 2020. [Online; accessed 04-October-2020].

[WL13] Yichao Wu and Yufeng Liu. Adaptively weighted large margin classifiers. *Journal of Computational and Graphical Statistics*, pages 416–432, 2013.

[WLJ07] Hansheng Wang, Guodong Li, and Guohua Jiang. Robust regression shrinkage and consistent variable selection through the lad-lasso. *Journal of Business & Economic Statistics*, pages 347–355, 2007.

[WM10] John Wright and Yi Ma. Dense error correction via l1-minimization. *IEEE Transactions on Information Theory*, pages 3540–3560, 2010.

[WMC+01] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for svms. In *Advances in neural information processing systems*, pages 668–674, 2001.

[WMM+10] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, pages 1031–1044, 2010.

[WN88] Laurence A. Wolsey and George L. Nemhauser. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1988.

[WN99] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

[WWL12] Lan Wang, Yichao Wu, and Runze Li. Quantile regression for analyzing heterogeneity in ultra-high dimension. *Journal of the American Statistical Association*, pages 214–222, 2012.

[WYG+08] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, pages 210–227, 2008.

[WZ14] Kuaini Wang and Ping Zhong. Robust non-convex least squares loss function for regression with outliers. *Knowledge-Based Systems*, pages 290–302, 2014.

[XCHP17] Guibiao Xu, Zheng Cao, Bao-Gang Hu, and Jose C Principe. Robust support vector machines based on the rescaled hinge loss function. *Pattern Recognition*, pages 139–148, 2017.

[XWX17] Yingchao Xiao, Huangang Wang, and Wenli Xu. Ramp loss based robust one-class svm. *Pattern Recognition Letters*, pages 15–20, 2017.

[YD19] Liming Yang and Hongwei Dong. Robust support vector machine with generalized quantile loss for classification and regression. *Applied Soft Computing*, page 105483, 2019.

[Yoh87] Victor J Yohai. High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics*, pages 642–656, 1987.

[YS09] Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing*. World Scientific, 2009.

[YTH14] Xiaowei Yang, Liangjun Tan, and Lifang He. A robust least squares support vector machine for regression and classification with noise. *Neurocomputing*, pages 41–52, 2014.

[YXW+10] Min Yang, Linli Xu, Martha White, Dale Schuurmans, and Yao-liang Yu. Relaxed clipping: A global training method for robust regression and classification. In *Advances in neural information processing systems*, pages 2532–2540, 2010.

[ZA05] Georgios Zioutas and Antonios Avramidis. Deleting outliers in robust regression with mixed integer programming. *Acta Mathematicae Applicatae Sinica*, pages 323–334, 2005.

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, pages 301–320, 2005.

[ZH+08] Cun-Hui Zhang, Jian Huang, et al. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, pages 1567–1594, 2008.

[Zou06] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, pages 1418–1429, 2006.

[ZY06] Peng Zhao and Bin Yu. On model selection consistency of lasso. *Journal of Machine learning research*, pages 2541–2563, 2006.