Variable selection in logistic regression using exact optimisation approaches

JV Venter

orcid.org / 0000-0002-2389-6255

Thesis accepted for the degree *Doctor of Philosophy in Science with Business Mathematics* at the North-West University

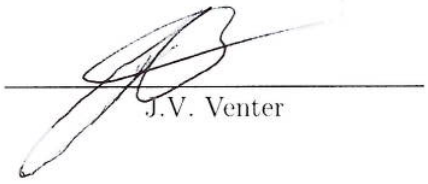
Promoter: Prof SE Terblanche

Graduation May 2020

22178384

Declaration

I, the undersigned, declare that the work contained in this thesis is my own work, except for references specifically indicated in the text, and that I have not previuosly submitted it elsewhere for degree purposes.



J.V. Venter

2019-11-04

Date

Abstract

Logistic regression modelling has been and still remains one of the most frequently used methods for the solving of binary classifcation problems, where the target variable of interest can take on one of two values. Futhermore, the logistic regression model formulation can also be extended to multi-class problems, where the response variable in question assumes more than two categorical levels. The extensive use of logistic regression models can most likely be attributed to various beneficial properties that these models exhibit over more advanced machine learning algorithms, such as their overall simplicity and their ability to produce descriptive end-solutions that can easily be deciphered. For this reason, logistic regression modelling is especially popular in application domains like medical research and the financial industry.

As is the case with most machine learning approaches and other statistical modelling techniques, variable selection is often required when developing a logistic regression model. In fact, in most problem settings the input dataset will consist of many potential predictor variables, where it is up to the modeller to find a suitable subset of these features which describes the problem in the most accurate and best possible manner. Obtaining a model that is based on a smaller set of inputs is generally considered as good practice and entails many benefits, such as the ability to yield interpretable final models and produce predictions that are more stable over time. Many variable selection techniques in logistic regression modelling applications exist, including computationally friendly approaches such as stepwise regression or penalised regression methods like the lasso and the elastic net. However, the work contained within this thesis is specifically directed towards the concept of best subset selection in regression modelling, which involves selecting a maximum of q variables from a total of p possible features in the input space and subsequently obtaining the most optimal q-variable model amongst all possible models consisting of q predictors. Best subset selection is much more resource intensive and time consuming than more conventional variable selection techniques, even for moderately sized datasets. However, it can produce mathematically proven optimal models.

In this thesis, a linearised approximation of the log-likelihood objective function is presented as a potential alternative to iterative fitting methods employed by logistic regression. The log-likelihood objective function is solved using linear programming techniques, such as the well-known simplex method. A modified version of the linearised logistic regression model is proposed,

which facilitates best subset variable selection. The resulting model is a mixed integer linear programming problem that incorporates a cardinality constraint on the number of variables. The suggested approach maintains many attractive properties, such as its ability to quantify the quality of the final variable selection solution, its independence of the subjective choice of p-values inherent to typical stepwise variable selection approaches and its capability to edge closer to optimality within increasingly reduced computing times when the correct settings are applied, even for large input datasets.

Computational results are presented to demonstrate the advantages of employing an exact mathematical programming approach towards variable selection in logistic regression applications. Empirical evidence suggests that the resulting model produces accurate and parsimonious solutions that are similar to or sometimes better than the benchmark, while still maintaining the beneficial properties listed above. Ultimately, the results documented in this thesis suggest that viable solutions can be obtained for hard optimisation problems, such as best subset selection, within appropriate time frames using an ordinary computer.

Keywords: logistic regression, linearisation, mixed integer linear programming, best subset selection

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to sincerely thank my PhD supervisor, Professor Fanie Terblanche. Thank you for all the late night and early morning phone calls, for granting me access to the university's computing infrastructure and software so that I can program my models and carry out my empirical experiments, for consistently booking out your diary so that we can sit and code together (whether it is at your office or at your home), for providing guidance on the adminstrative burdens related to part time studies and for just being there throughout every step of my journey. Completing a PhD part time while still having a full time job and family at home can certainly become overwhelming at times and often leads to intermittent slumps and periods of discouragement. However, you were always just a text message or call away and managed to pick me up every time. You have the ability to give me a motivational boost after each and every consultation session or phone conversation. We as students like to think that we "suffer" the most during the course of our studies, however, we often forget how much time, energy and effort are spent by our supervisors on us. And for that, I truely commend you. You started out as my supervisor, but became a mentor and trusted friend along the way. You will always be known as one of the key people that had a significant impact on my journey in life.

Next, I would like to thank my work colleagues at First National Bank South Africa for supporting my PhD. I am very grateful that I have the privilege to work in an environment where consistent self-improvement and further education is greatly encouraged and forms part of our organisational structure.

Lastly, I would like to extend a tremendous amount of gratitude towards my loving wife Toni. Thank you for remaining by my side every step of the way, for putting up with me being locked in my study during week nights and over weekends, for missing out on so many fun activities and family events so that we can stay home in order for me to do some research and for supporting me all these years. Thank you for realising that my PhD is a very precious and personal goal that I have set for myself and for believing in my ability to always come out on top and to set myself apart from the rest. I could not have done this without you.

List of Abbreviations

AUC	Area Under the Curve
BBA	Branch-and-bound Algorithm
FN	False Negative
FP	False Positive
IP	Integer Programming
LASSO	Least Absolute Shrinkage and Selection Operator
LBA	Leaps and Bounds Algorithm
LogL	Log-likelihood
LP	Linear Programming
MCP	Minimax Concave Penalty
MLE	Maximum Likelihood Estimate
MIO	Mixed Integer Optimisation
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
MIQP	Mixed Integer Quadratic Programming
NP	Nondeterministic Polynomial Time Problems
OLS	Ordinary Least Squares
SCAD	Smoothly Clipped Absolute Deviation
SSE	Sum of Squared Errors

SSTO	Total Sums of Squares
TN	True Negative
TP	True Positive

Chapter 1

Introduction

Since the 20th century, logistic regression models have garnered an immense following in the area of predictive modelling and have become highly regarded amongst statisticians, academics and data scientists alike. Binary regression modelling still remains one of the most frequently applied techniques in a variety of present-day application domains. More specifically, it has gained considerable popularity in the finance industry due to a number of advantages associated with the technique, such as its ease of implementation and interpretability (even by stakeholders who are not mathematically inclined) and enabling its users to easily monitor the stability of predictor variables over time (a procedure that is paramount in scorecard modelling). In fact, logistic regression is still the most commonly applied modelling approach during the development of scorecards that are used to classify and predict potential future delinquencies when extending credit (Siddiq, 2006).

When regression models are being constructed, it is often reasonable to assume that the estimated regression parameter vector is sparse and contains many zeros. This concept is broadly referred to as *variable selection* and it implies that only a handful of predictor variables in the modelling dataset or design matrix will have an effect on future predictions, instead of including *all* of the features associated with each observation in the final model. The inclusion of a limited number of variables in the end-solution entails many benefits, such as yielding descriptive models which can easily be interpreted and dissected. A model with fewer variables will also tend to be more stable and less volatile with its predictions. Although computability is generally not a concern when fitting logistic regression models, the development of parsimonious models through the application of certain variable selection techniques, such as best subset selection, does pose some challenges. In spite of advances in computing technology over the last decade, practical experience has shown that best subset selection remains an extremely resource intensive variable selection method, even for a modest number of predictors. According to Potts and Patetta (1999), best subset selection becomes computationally infeasible when considering datasets that contain more than approximately 40 to 50 inputs. When using the software package SAS, Lund

(2017) states that analysts should avoid best subset selection when the number of variables is 75 or more, due to the exponential increase in execution time. Specifically, best subset selection is considered to be an NP-hard problem (Natarajan, 1995). NP problems refer to a set of mathematical optimisation problems where it is theoretically possible to evaluate the solution of the problem in polynomial time, but it is very difficult to arrive at the solution itself. NP-hard problems are seen as the hardest problems that exist in NP. Most NP problems are notoriously difficult to solve, even by modern software packages. Many practical alternatives to best subset selection exist, such as stepwise regression approaches (which include forward and backward selection) or penalised regression techniques like as lasso (Tibshirani, 1996) and the elastic net (Zou and Hastie, 2005). These methods are more commonly used in practice due to their computational friendliness, however, it may be to the detriment of optimality.

The solving of large scale and real-world problems using exact approaches have been neglected for a considerable amount of time within the statistical community due to a widespread belief that such methods may be intractable (Bertsimas and King, 2016). Nevertheless, in spite of the arithmetical challenges faced within the realm of NP problems, significant improvements in computing power and algorithmic advances over the last three decades have resulted in an incredible 200 billion factor speedup in the solving of hard optimisation problems – see Bertsimas et al. (2016). As a result, gradual interest in the application of exact mathematical modelling approaches have started to resurface over recent years. Authors such as Bertsimas et al. (2016) and Bertsimas and King (2016) have done commendable work in this regard by composing a mixed integer quadratic optimisation problem formulation (MIQP) that allows for the design of linear regression models that facilitate variable selection via best subset selection. The authors show that their suggested model has the ability to yield accurate results within an acceptable amount of time for reasonably sized datasets. By following an approach that has algorithmic similarities to Bertsimas et al. (2016), Maldonado et al. (2014) proposes a SVM (support vector machine) model with best subset feature selection as a mixed integer linear programming problem (MILP). While the aformentioned sets of authors focus on linear regression or SVM's, Sato et al. (2015) addresses logistic regression problems by devising a linearised approximation of the logistic loss function. The resulting model is presented as a MILP and performs automatic variable selection by including a penalty term in the objective function. More recently, a MILP formulation for multinomial regression models was put forth by Kamiya et al. (2019), which attempts to fit categorical regression models (instead of binary models like with logistic regression) within a mixed integer programming context.

In this thesis, the use of exact mathematical modelling approaches is suggested as an alternative to traditional variable selection applications, such as stepwise regression or penalised models. A linearised approximation of the log-likelihood function for binary regression problems, where the target vector $\mathbf{Y} \in \{0, 1\}$, is first introduced as a linear programming problem (LP).

Next, the logistic regression LP is extended to allow variable selection via the application of best subset selection by presenting the model as a MILP. This is achieved by the inclusion of integer choice variables in the LP model constraints.

1.1 Objectives

Firstly, the goal is to present the objective function in logistic regression as a linear programming problem by proposing a linearised approximation for the log-likelihood function. Noting that the log-likelihood objective function is nonlinear, a linearised approach is pursued for three main reasons, namely:

- Many commercial software packages and solvers still struggle to handle mixed integer nonlinear programming problems due to their complexity and tendency to be computationally expensive (Brandimarte, 2006). Alternatively, most solvers can accommodate linear programming problems quite easily.
- Nonlinearity can often result in numerical instabilities during computation and model fitting (Sato et al., 2015).
- If formulated correctly, a linear programming problem (LP) has the ability to yield a local optimal solution which is also a global optimum.

The main objective of this thesis is, therefore, to allow the aforementioned logistic regression LP to perform best subset selection by including integer choice variables in the constraints, thereby transforming it into a mixed integer linear programming problem. When compared with other variable selection techniques, the resulting MILP can be associated with the following major benefits:

- It has the ability to quantify the quality of the final logistic regression model by providing the user with a guarantee on optimality (this is a feature of the algorithmic approach that is utilised for solving MILPs).
- It is entirely independent of p-values, which are used in stepwise selection procedures to determine the significance of a predictor variable and to decide on its inclusion in the final model. Many drawbacks associated with this approach have been well documented in the literature.
- It is an underestimator for the MLE achieved by the nonlinear log-likelihood objective function and therefore serves as a lower bound for the optimal solution. This means that the user never has to accept any model with a weaker solution than the one produced by the MILP and that model fit can only be improved going forward.

Lastly – and perhaps most importantly – the aim would be to show that the MILP formulation can yield **accurate** and **parsimonious** models within a **reasonable amount of time**. By doing so, it is demonstrated that viable solutions for NP-hard problems, such as best subset selection, is attainable for noticably large modelling sets, even when standard computing technology is used to solve them.

A condensed version of the work contained within this thesis can be found in Venter and Terblanche (2019). After considering the logistic regression MILP formulation and extensive empirical evidence presented by the authors, it can be shown that the objectives listed in this section can be achieved with reasonable success.

1.2 Outline

The layout of this thesis is structured as follows: in Chapter 2 a brief history on logistic regression modelling is provided, followed by the derivation of the log-likelihood objective function and a detailed explanation of the commonly used Newton-Raphson method for fitting logistic regression models to a set of data. Chapter 3 furnishes the reader with a thorough background on the concept of best subset selection, along with discussions on a variety of variable selection techniques that are frequently utilised during the development of regression models. These include stepwise approaches as well as penalised regression methods like ridge regression, the lasso, the elastic net and minimax concave (MCP) and smoothly clipped absolute deviation (SCAD) penalties. Chapter 4 introduces concepts on exact mathematical modelling, such as linear programming problems and mixed integer programming problems. This chapter also contains a substantial amount of mathematical rigour for the purpose of communicating the properties inherent to these mathematical models clearly. In Chapter 5, a linearised approximation of the nonlinear log-likelihood function in logistic regression is proposed as an LP. The model is then subsequently expanded to include binary integer variables in its constraints, thereby transforming it into a MILP. The resulting MILP allows the model to perform best subset selection while simultaneously estimating its regression coefficients. The suggested linearised model is also compared with existing model formulations and key similarities and differences that exist between methodologies are discussed. Chapter 6 contains computational results obtained from fitting models to both simulated and real-world data using the LP and MILP formulations proposed in Chapter 5 and the subsequent evaluation thereof. In Chapter 7, experimental evidence is presented which allows the reader to analyse the trade-off between the quality of solutions obtained and model execution time given different levels of granularity imposed on the linearised approximation of the log-likelihood function. Lastly, in Chapter 8, a set of summary remarks and discussion around future work concludes the thesis.

Additional appendices also provide the reader with supplementary information on a select

few topics that are touched on during the course of the thesis. Appendix A provides more detail on the likelihood ratio test for selecting significant predictor variables in logistic regression during stepwise modelling, while Appendix B contains information on the inner workings of the ROC or Receiver Operator Characteristic curve. In Appendix C comprehensive information can be found on branch-and-bound techniques used to perform best subset selection in software packages such as SAS.

Chapter 2

Logistic regression

2.1 Introduction

Logistic regression models form part of a class of regression methods used to model a discriminant function which is aimed at solving problems where the dependent variable vector $\mathbf{Y}$ is categorical. Specifically, each entry Y_i can take on a value in a discrete set G that contains K classes labelled $1, 2, \ldots, K$. The equation used to express logistic regression models may seem very similar to that of a linear regression model where the output vector $\mathbf{Y}$ is continuous, or $\mathbf{Y} \in \mathbb{R}$. However, the assumptions made and the methods used during the fitting of dichotomous models differ quite significantly from those utilised for linear regression.

Hastie et al. (2001) addresses problems where the target assumes values within a discrete set, resulting in an input space that can be divided into regions which are labelled based on prespecified classifications. The boundaries of these regions can be smooth or rough, depending on the method of classification used. But, for an important class of predictors, these decision boundaries are linear, which constitutes the main focus of this chapter. Given K classes, the fitted model for the k-th indicator response is given by $f_k(x) = \hat{\beta}_{k0} + \hat{\beta}_k^T x$. The decision boundary between class k and any other class l is the set of points for which $f_k(x) = f_l(x)$, which is an affine set or hyperplane separating the two regions. This is true for all classes, meaning that the decision region is divided by sets of piecewise hyperplanes or decision boundaries. Specifically, regression methods model a discriminant function $\delta_k(x)$ for each class and subsequently classifies x to the class with the largest output value. Obtaining the posterior probabilities $P(Y_i = 1|\mathbf{X}_i = x)$ is henceforth of great interest within these problem settings. Therefore, if either $\delta_k(x)$ or $P(Y_i = 1|\mathbf{X}_i = x)$ are linear in x, the decision boundaries will also be linear. Ultimately, a monotone transformation of $\delta_k(x)$ or $P(Y_i = 1|\mathbf{X}_i = x)$ is required in order for the decision boundaries to be linear. As seen in Section 2.3, the use of logistic regression modelling produces the desired discriminant function, while also providing the posterior probability of interest as output and maintaining various attractive properties.

The derivation of the logistic regression model and the traditional Newton-Raphson method used to fit such a model to a set of data is discussed in this chapter. However, a brief history on logistic regression and its widespread popularity is presented first.

2.2 A brief history on logistic regression and its importance in present-day applications

Many authors, such as Cramer (2002) and Wilson and Lorenz (2015), regard the informal formulations produced in the 19th century by astronomer-turned-statistician Quetelet (1795 - 1874) and his pupil Verhulst (1804 - 1849) as the birth of the logistic regression model. In an attempt to model the growth of populations, the time path of $W(t)$ and its growth rate is represented by

$$\dot{W}(t) = dW(t)/dt,$$

where $\dot{W}(t)$ is assumed to be proportional to $W(t)$, meaning that

$$\dot{W}(t) = \beta W(t),$$

with $\beta = \dot{W}(t)/W(t)$, where β is a constant rate of growth. Consequently, this leads to the exponential growth model given by

$$W(t) = Ae^{\beta t},$$

where A is in some cases assumed to be $W(0)$. Verhulst added an extra term to the growth model which represented a resistance to further growth, formulating it as

$$\dot{W}(t) = \beta W(t) - \phi(W(t)).$$

When the above equation is quadratic, a logistic model is obtained. The resulting logistic model can be written as

$$\dot{W}(t) = \beta W(t)(\Omega - W(t)),$$

where Ω denotes the upper bound of W. This implies that growth is proportional to the population encapsulated by $W(t)$ and that the remaining room left for expansion is given by $\Omega - W(t)$. By expressing $W(t)$ as a proportion $P(t) = W(t)/\Omega$, the following equation is obtained:

$$P(t) = \beta P(t)[1 - P(t)].$$

This leads to a solution for the differential equation which can be written as

$$P(t) = \frac{\exp(\alpha + \beta t)}{1 + \exp(\alpha + \beta t)}.$$

The value $P(t)$ above was dubbed as the logistic function by Verhulst. $W(t)$ is then given by

$$W(t) = \Omega \frac{\exp(\alpha + \beta t)}{1 + \exp(\alpha + \beta t)}.$$

It should be noted that the differential equation presented for $P(t)$ above is identical to the inequality used to obtain the posterior probabilities $P(Y_i = 1|x = \mathbf{X}_i) = \pi_i$ in a logistic regression model, which will be discussed shortly in Section 2.3.

The work performed by Quetelet and Verhulst was neglected for a considerable while, until it resurfaced roughly 70 years later shortly after the end of World War I. In 1929, researchers such as mathematician Lowell Reed and the Director of Biometry and Vital Statistics at John Hopkins University, Raymond Pearl, produced papers wherein logistic functions are applied to the food needs of a growing population and to autocatalytic reactions – see Reed and Berkson (1929). Furthermore, authors such as Wilson (1929) and Winsor (1932) documented some of the properties inherent to the logistic distribution function and its resemblance to the normal distribution.

Shortly after the work performed by the aforementioned authors, Gaddum (1933) and Bliss (1934) are credited for the formulation of the probit model. As mentioned later in Section 2.3, the probit regression model bears a striking resemblance to the logistic regression model, with the main difference being that the logistic model assumes a logistic density for the error terms of some hidden, latent continuous variable on which the target is based, whereas the probit regression model assumes that the errors have a normal underlying distribution. Note that this was done in a time where the assumption of normality was commonplace and where predefined formulas and tables existed for deriving test statistics from the normal distribution in the absence of computers. Further publications by Bliss introduced the first use of the term probit, short for "probability unit". Additionally, Bliss is also credited with the first formal derivation of the very important maximum likelihood estimation of the probit curve (Bliss, 1935).

The use of a logistic curve as a potential alternative to the normal probability function is first introduced by Joseph Berkson, co-author to some of Reeds papers, in 1944 – see Berkson (1944). Berkson also proposed the use of the term "logit" as an analogy to Bliss' probit. The logistic function, first considered in the field of bioassay, presented itself as a model that can potentially be extended to regression applications where binary outcomes are related to a set of inputs and where the researcher was not necessarily required to have a theoretical background of the problem. At the time of its invention, the logit was widely disputed and regarded as

inferior to the probit. However, the practical ease of computation that the logit had over the probit, especially in maximum likelihood estimation, resulted in the logit enjoying increasingly more popularity amongst academics and gradually becoming more commonplace in practice.

By 1970, the logit had become a noteworthy alternative to the probit. Academics came to realise that its analytical properties allowed for much broader applications – beyond that of bioassay and medical fields – such as discriminant analyses, log-linear models and case-controlled studies. The logistic curve also garnered more attention due to the analytical advantages inherent to the logit transformation and its ability to accommodate binary dependent variables during a time when discrete outcomes gained considerable traction in statistical discourse. The attractiveness of the logistic model was supported and pioneered by academics such as statistician David Cox, who published several papers on the matter during the 1960's, e.g. Cox (1969). A short while later, the application of the logistic model within discriminant analysis was formally recognised alongside its unique proximity to log-linear models (Bishop et al., 1975).

Attempts towards the generalisation of logistic regression to multinomial cases are explored in Gurland et al. (1960), Mantel (1966) and Theil (1969). The first link between a multinomial logit and the theory of discrete choice was established by McFadden (1973), which served as a theoretical foundation for the logistic model. The work produced by McFadden earned the author the Nobel Prize in 2000. The first explicit formulation of a regression model that deals with a discrete target was produced by McKelvey and Zavoina (1975). The authors utilised a probit distribution in an attempt to model voting behaviour of US congressman and solved a problem wherein binary outcomes were linked to a set of covariates. It is argued by Cramer (2002) that the wider acceptance of probit and logistic regression was greatly assisted by the invention of the modern-day computer and the introduction of faster and more efficient algorithms for dealing with maximum likelihood estimation.

For a much more comprehensive background on the history of logistic regression, the reader is referred to Cramer (2002) and Wilson and Lorenz (2015).

Over the years, logistic regression has enjoyed considerable popularity within medical and financial research domains and still remains one of the most used supervised modelling techniques in the South African banking industry. The use of logistic regression modelling is especially paramount to credit scoring and the development of scorecard models. Briefly stated, credit scoring involves the assessment of risk of a particular individual or applicant. This includes the assignment of scores related to characteristics that represent the debt of the borrower, historical default tendencies and other losses experienced in the past as an indication of the level of risk associated with the borrower. Ultimately, the aim of the scorecard is to produce a single aggregated risk indicator based on a set of risk factors (Bolton, 2009). Historically, these decisions were largely judgemental in nature and involved the approval of a bank manager who would assess the risk associated with a customer based on personal knowledge of the applicant,

good faith and an overall inclination towards the risk appetite of the lender. However, this method was marred with many shortcomings. Approval rates could differ wildly on a daily basis (for example, based on the manager's mood or different managers making different decisions), the process could rarely be replicated at different branches, knowledge pertaining to how a decision was made was difficult to pass on to other staff, bank managers could not handle large volumes of applications in a short period of time and, overall, the process was clearly not objective. Indeed, as banking regulations became more strict, a new method for approving the granting of credit had to be instituted. For example, banking institutions in South Africa, which are governed by the South African Reserve Bank, are not permitted to base their decisions on a variety of demographic factors relating to a customer, such as religion, race, gender, language, societal status and various other inputs. This notion is completely contradictory to the aforementioned subjective manner in which bank managers used to grant credit. Anderson (2007) notes that the first retail credit scorecard was proposed in the United States in 1941 and was based on an applicant's job status, number of years at current employer, number of years at current address, details surrounding life insurance policies, gender and the monthly instalment of the requested credit. Furthermore, the increasing use of credit cards warranted a decision engine that radically reduced decision time. Myers and Forgy (1963) were some of the first authors to propose the application of multivariate discriminant analysis in credit scoring. Finally, in 1975, credit scoring was accepted as a whole in the USA in order to comply with the US Equal Credit Opportunity Act I. The application of scorecards in the United Kingdom also became standard practice after the first UK retail credit card was launced roughly a decade earlier in 1966.

In the South African retail banking sector, logistic regression still remains the dominant method for designing credit scorecards. A thorough discussion on the process for developing credit scorecards by means of logistic regression models is provided by Siddiq (2006) – a piece of literature that has become a common sight on many South African scorecard developers' desks. Logistic regression modelling is also the preferred technique utilised in the design of probability of default (PD) models used in the estimation of the expected loss of a particular portfolio in the bank. The aforementioned expected loss (EL) models enable the bank to determine the amount of capital to set aside as provision for potential losses that the institution may suffer due to an increase in credit risk [1] and is required by banking regulations. This includes IAS39 and the recently introduced IFRS9 models. The sign-off and approval of scorecard and capital and provisioning models by a bank's executive committee, as well as approval from the bank's audit committee, depends on a variety of success factors. Some of these factors touch on important aspects such as the transparency and stability of the models – two advantages inherent to logistic regression models that are not always easily proven for many other supervised learning techniques.

[1] Risk associated with the inability to recover payments of borrowers.

Indeed, logistic regression models maintain their popularity due to intrinsic beneficial characteristics, such as:

- Producing parsimonious models that still retain suitable levels of accuracy.
- The ability to easily monitor and report on the stability of the variables in the model over time.
- The absence of complex algorithms and requiring less computational resources when fitting models to data.
- Producing models that are easy to understand and interpret, even by stakeholders and decision makers who are not mathematically inclined.
- The availability of a plethora of literature, research and use cases that document the models' proven success, as opposed to more modern machine learning algorithms that have only recently come to the fore.

Following the success and widespread adoption of these binary models in credit scoring and default prediction, logistic regression models have also been successfully extended to many other areas that relate to dichotomous problems, such as attrition modelling (see e.g. Oghojafor et al. (2012)), fraud detection (see e.g. Sahin and Ekrem (2011)) and propensity modelling (see e.g. Gant and Crowland (2017)).

Unbeknownst to many, logistic regression models are also integral to some of the more modernised machine learning algorithms that have become increasingly popular in recent times. In feed-forward neural networks, the logistic regression model is a commonly used activation function in hidden layers, the results of which are used as inputs in subsequent layers. The output layer of the neural network may also utilise a logistic activation function (Walsh, 2002). Gradient boosting models are iterative machine learning algorithms wherein predictive models are fitted during each step in an attempt to reduce the error produced by the model in the previous step. Popular models utilised in individual steps include decision "stumps" [2] or low-dimensional regression models (Friedman, 2001). Boosting algorithms that utilise logistic regression models are also referred to as additive logistic regression (Friedman et al., 2000).

2.3 The logistic regression model

For purposes of the model derivations presented in this section, assume that the target vector $\mathbf{Y}$ is binary, i.e. the set G only contains $K = 2$ classes. Kutner et al. (2005) provides a thorough

[2] Decision stumps are shallow decision trees that do not contain many branches.

and easily interpretable explanation on obtaining the discriminant function $\delta_k(x)$ in logistic regression, which is discussed next. Consider the simple linear regression model

$$Y_i^c = \mathbf{X}_i^T \beta^c + \varepsilon_i^c \quad \text{for } i = 1, \ldots, n, \tag{2.1}$$

where Y_i^c is a continuous latent variable, β^c is a $p \times 1$ parameter vector of regression coefficients and $\mathbf{X}_i$ is a $p \times 1$ vector of input variable values for the i-th observation in a dataset consisting of n records. Assume that the error terms ε_i^c are independent, follow a certain distribution with standard deviation σ_c and have an expected value of zero. Suppose that a binary response vector $\mathbf{Y}$ is created such that

$$Y_i = \begin{cases} 0 & \text{if } Y_i^c > k_c, \\ 1 & \text{if } Y_i^c \leq k_c, \end{cases}$$

where k_c is a some scalar value.

Now, assume that Y_i is a Bernoulli random variable with distribution $P(Y_i = 1) = \pi_i$ and $P(Y_i = 0) = 1 - \pi_i$. Ultimately, since $E(Y_i) = 1(\pi_i) + 0(1 - \pi_i) = \pi_i$, it follows that

$$E(Y_i) = \pi_i.$$

Since the output of interest is a probability, the model requires a response function that constrains the mean response $E(Y_i)$ such that it lies between 0 and 1, or $0 \leq E(Y) = \pi \leq 1$. In the case of straightforward linear regression, where $\mathbf{Y} \in \mathbb{R}$, the objective function being solved for is given by $\min \frac{1}{2} \sum_{i=1}^{n} (Y_i - \mathbf{X}_i^T \beta)^2$, which can be written as $\min_{\beta} \frac{1}{2} \|\mathbf{Y} - \mathbf{X}^T \beta\|_2^2$, which minimises the sum of the squared residuals of the model. Clearly, such a linear response function will prove to be inadequate, since the results may fall outside of the limits imposed on a model that should produce posterior probabilities as output. A possible solution to this could be the use of weighted least squares for unequal error variances. However, for larger sample sizes this method yields estimators that are asymptotically normal under general conditions, even if the distribution of the error terms is not normal at all. Ultimately, the need for mean responses to be located between 0 and 1 will most likely rule out a linear response function in all instances. For example, suppose that a relationship is modelled between $Y_i \in \{0, 1\}$ and a single predictor variable X, where larger values of X will result in a higher likelihood of having $Y_i = 1$ and vice versa. The use of a linear model that is subjected to the aforementioned constraints might require a probability of 0 as the mean response for all observations with smaller values of X and a probability of 1 as the mean response for all observations with large entries for X. Such a model will more often than not be considered as unreasonable. Instead, a model where probabilities between 0 and 1 are reached asymptotically is desired. Figure 2.1 graphically illustrates the difference between a linear response function and one where the probabilities are achieved asymptotically.

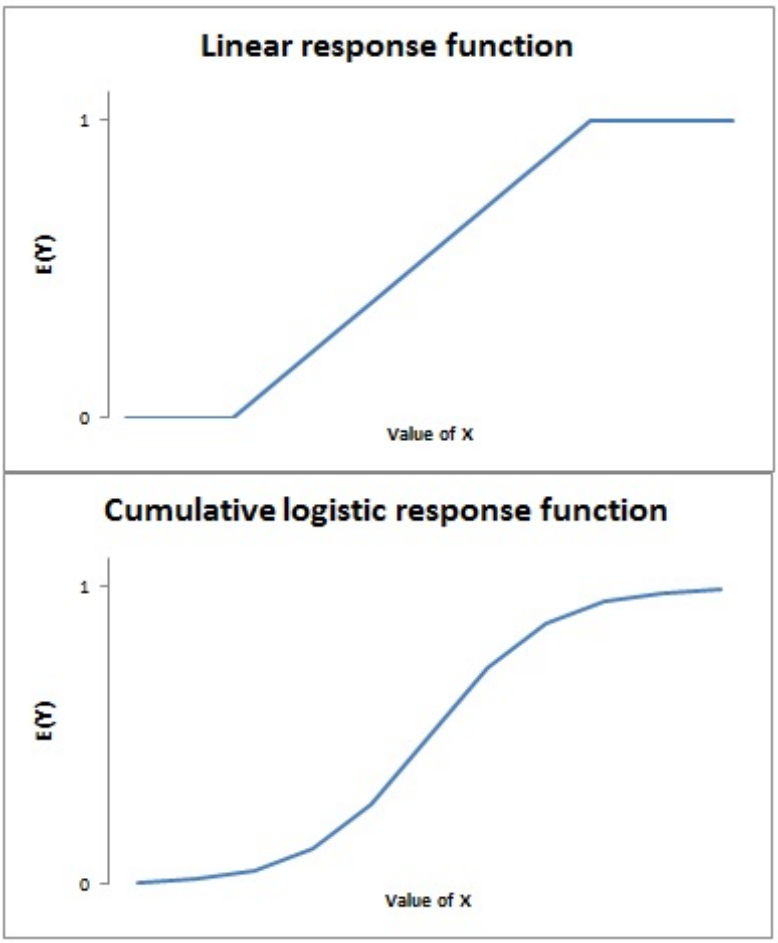


Figure 2.1: Linear vs logistic response function

Suppose that the error terms in (2.1) follow a logistic distribution[3]. The density of a logistic random variable ε with mean zero and standard deviation $\sigma = \pi/\sqrt{3}$ is given by

$$f(\varepsilon) = \frac{\exp(\varepsilon)}{(1+\exp(\varepsilon))^2}.$$

The variable ε has the following cumulative distribution function:

$$F(\varepsilon) = \frac{\exp(\varepsilon)}{1+\exp(\varepsilon)}. \tag{2.2}$$

If ε_i^c from (2.1) has a mean of zero and logistic density with standard deviation σ_c, it follows that

[3]The logistic density is very similar to the normal density, except for the fact that the logistic distribution has slightly heavier tails.

$$P(Y_i = 1) = E(Y_i) = \pi_i = P(Y_i^c \leq k_c), \tag{2.3}$$
$$= P(\frac{\varepsilon_i^c}{\sigma_c} \leq \mathbf{X}_i^T \beta^*) \tag{2.4}$$
$$= P(\frac{\pi}{\sqrt{3}}\frac{\varepsilon_i^c}{\sigma_c} \leq \frac{\pi}{\sqrt{3}}\mathbf{X}_i^T \beta^*) \tag{2.5}$$
$$= P(\varepsilon \leq \beta^T \mathbf{X}_i) \tag{2.6}$$
$$= F(\beta^T \mathbf{X}_i), \tag{2.7}$$

where $\beta_0^* = \frac{(k_c - \beta_0^c)}{\sigma_c}$, $\beta_l^* = \frac{-\beta_l^c}{\sigma_c}$ for $l = 1, \ldots, p$ and β is a vector of logistic regression parameters. This leads to

$$P(Y_i = 1) = \frac{\exp(\beta^T \mathbf{X}_i)}{1 + \exp(\beta^T \mathbf{X}_i)}. \tag{2.8}$$

Appying the inverse of the cumulative distribution function in (2.2) yields

$$F^{-1}(\pi_i) = \beta^T \mathbf{X}_i = \log(\frac{\pi_i}{1 - \pi_i}). \tag{2.9}$$

where the ratio $\frac{\pi_i}{1-\pi_i}$ is called the odds and the predictor in (2.9) is known as the logit response function, which is the monotone transformation of the discriminant function $\delta_k(x)$. Essentially, the regression line models the *log of the odds* . The above derivation can easily be extended to probit regression by assuming a normal distribution for ε_i^c in (2.1) instead of a logistic distribution.

The logistic mean response function has the following properties:

- It is bounded between 0 and 1 and reaches its limits asymptotically.
- Changing the sign of any regression coefficient, β_l, changes the mean response from a monotone increasing to a monotone decreasing function on the axis of variable $\mathbf{X}_l$.
- The response function has a symmetry property. Recoding the target variable $\mathbf{Y}$ such that the 0's become 1's and the 1's become 0's results in the signs of all the regression coefficients being reversed. However, the posterior probability obtained as output remains the same. This follows from the symmetry of the logistic distribution, since $F_L(x) = 1 - F_L(-x)$, which implies that $1 - F_L(\beta^T \mathbf{X}_i) = F_L(-\beta^T \mathbf{X}_i)$.

The logistic regression model is fitted to the data using the method of maximum likelihood (which is different from minimising the sum of the squared residuals when estimating linear regression models). The resulting fit will produce parameter estimates for the regression coefficients in the logistic response function. Recall that the observations Y_i are assumed to be

independent and follow a Bernoulli distribution with $P(Y_i = 1) = \pi_i$ and $P(Y_i = 0) = 1 - \pi_i$. The probability distribution of Y_i is given by

$$f_i(Y_i) = \pi_i^{Y_i}(1 - \pi_i)^{1-Y_i}.$$

Clearly, if $Y_i = 1$ then $f_i(Y_i) = \pi_i$, whereas $Y_i = 0$ results in $f_i(Y_i) = 1 - \pi_i$. Given that the observations are independent, the joint distribution function can be written as

$$L(Y_1, \ldots, Y_n) = \prod_{i=1}^{n} f_i(Y_i) = \prod_{i=1}^{n} \pi_i^{Y_i}(1 - \pi_i)^{1-Y_i}. \tag{2.10}$$

Taking the logarithm of (2.10) reveals

$$\begin{aligned}
\log L(Y_1, \ldots, Y_n) &= \log \left[\prod_{i=1}^{n} \pi_i^{Y_i}(1 - \pi_i)^{1-Y_i} \right] \\
&= \sum_{i=1}^{n} Y_i \log(\pi_i) + (1 - Y_i) \log(1 - \pi_i) \\
&= \sum_{i=1}^{n} Y_i \log \left(\frac{\pi_i}{1 - \pi_i} \right) + \sum_{i=1}^{n} \log(1 - \pi_i).
\end{aligned}$$

Since $E(Y_i) = \pi_i$, it follows that

$$1 - \pi_i = \frac{1}{1 + \exp(\beta^T \mathbf{X}_i)}.$$

Additionally, (2.9) shows that

$$\log \left(\frac{\pi_i}{1 - \pi_i} \right) = \beta^T \mathbf{X}_i.$$

Ultimately, this leads to the log-likelihood function:

$$\log L(Y_1, \ldots, Y_n) = \sum_{i=1}^{n} Y_i(\beta^T \mathbf{X}_i) - \sum_{i=1}^{n} \log[1 + \exp(\beta^T \mathbf{X}_i)]. \tag{2.11}$$

By maximising (2.11), the optimum solution values for the vector of regression parameters are obtained by essentially allowing the model to choose estimates for the coefficients that are most consistent with the training data.

2.4 Fitting logistic regression models

One of the most popular methods used for estimating regression coefficients in logistic regression is the iterative Newton-Raphson method (Okeh and Oyeka, 2013). Many software packages employ this method in one way or another. A concise and accurate outline for fitting binary regression models by utilising the Newton-Raphson procedure is provided by Hastie et al. (2001). Suppose that the design matrix $\mathbf{X}$ contains p predictor variables and includes a column vector consisting of 1's to accommodate the intercept term of the model. Finding the first derivatives of equation (2.11) and setting equal to zero yields the following score equations:

$$\frac{\partial \log L(\beta)}{\partial \beta} = \sum_{i=1}^{n} \mathbf{X}_i (Y_i - \pi_i) = 0. \tag{2.12}$$

Equation (2.12) will result in $p+1$ inequalities (one for each column of the design matrix) that are nonlinear in β. In order to apply the Newton-Raphson method, the second order derivatives or *Hessian matrix* is also needed, which can be expressed as

$$\frac{\partial^2 \log L(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{n} \mathbf{X}_i \mathbf{X}_i^T \pi_i (1 - \pi_i). \tag{2.13}$$

During the first iteration, $it = 0$, the procedure starts off with an initial guess for the vector of regression coefficients, after which each update is given by

$$\beta^{it+1} = \beta^{it} - \left[\frac{\partial^2 \log L(\beta)}{\partial \beta \partial \beta^T} \right]^{-1} \left(\frac{\partial \log L(\beta)}{\partial \beta} \right), \tag{2.14}$$

where the derivatives in (2.14) are evaluated at β^{it}. The second part of (2.14) involving the first and second order derivatives is known as the step size.

In matrix notation, (2.12) and (2.13) can be written as

$$\frac{\partial \log L(\beta)}{\partial \beta} = \mathbf{X}^T (\mathbf{Y} - \mathbf{\Pi}), \qquad \frac{\partial^2 \log L(\beta)}{\partial \beta \partial \beta^T} = -\mathbf{X}^T \mathbf{W} \mathbf{X},$$

where $\mathbf{\Pi}$ is a $n \times 1$ vector of estimated posterior probabilities, where the i-th element is equal to π_i and can be obtained by evaluating the cumulative logistic distribution function at β^{it}. The $n \times n$ diagonal matrix $\mathbf{W}$ has off-diagonal elements equal to $\pi_i(1 - \pi_i)$ with all other entries being exactly zero. The step outlined in (2.14) can then be expressed as

$$\beta^{it+1} = \beta^{it} + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{Y} - \mathbf{\Pi}),$$

which can be rewritten as

$$\beta^{it+1} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}[\mathbf{X}\beta^{it} + \mathbf{W}^{-1}(\mathbf{Y} - \mathbf{\Pi})]. \tag{2.15}$$

These equations are then solved iteratively until convergence is achieved where $\log L(\beta^{it+1}) - \log L(\beta^{it}) \leq \epsilon$ for some very small value ϵ.

The Newton-Raphson procedure is a *locally convergent method*, meaning that iteratively solving the steps given in (2.14) and (2.15) will result in convergance if the intial guess for the parameter vector is close to the root. Yet, overshooting may occur and convergence is never guaranteed (Hastie et al., 2001). A possible solution to overshooting is a useful remedy known as *step-halving* or *ridging*, which involves reducing the step size in (2.14) in an attempt to force the algorithm closer towards the optimum solution. Many popular software packages, such as SAS, employ variations of step-halving when the procedure discovers that the likelihood evaluated at the current iteration is less than the likelihood obtained during the previous iteration. However, it is important to note that the log-likelihood function is a *concave* function (the concepts of concave and convex functions are discussed in Chapter 4). This implies that convergence is typically achieved and that the Newton-Raphson procedure will most likely produce an optimal solution for the estimated coefficient vector $\hat{\beta}$.

At this point, the inner workings of a logistic regression model have been thorougly introduced, which includes the derivation of the log-likelihood objective function being maximised during model execution and the explanation of the popular Newton-Raphson method used to find a solution for the maximum likelihood estimate. However, the modeller also has to consider the properties of the final solution produced for the estimated regression coefficient vector $\hat{\beta}$, which, in turn, has a strong dependency on the relationship between the input variables and the response and the predictors chosen for inclusion in the final model. Chapter 3 explores various methods used for selecting the most approriate predictor variables when estimating regression parameters during model fitting.

Chapter 3

Variable selection in logistic regression models

3.1 Introduction

In order to conduct meaningful statistical inference and to obtain parsimonious models that are easily interpretable, it is often reasonable to assume that the set of true regression coefficients β might be approximated by a sparse vector $\hat{\beta}$ that contains many zeros. This implies that only a subset of q predictors out of the possible p inputs, with $q \leq p$, will have an effect on the response in the regression model. In this chapter, the concept of best subset selection is first introduced, since the best subset selection problem will form the basis of all the work that has been carried out in subsequent chapters. Afterwards, various alternative and popular variable selection techniques which are commonly utilised when fitting regression models, are presented. These include stepwise methods and variable selection via regularisation. Regularised models facilitate variable selection by introducing penalty terms into the objective function of the regression model.

It should be noted that many variable selection techniques, such as lasso, ridge regression and the elastic net, are explained in terms of linear regression models by most of the authors. However, whether the dependent variable is conintuous, $\mathbf{Y} \in \mathbb{R}$, or binary, $\mathbf{Y} \in \{0, 1\}$, is of little concern, as the same terminology is usually applied in both settings. As shown in Chapter 8 of Tibshirani (1996), consider any model indexed by a parameter vector β which is estimated by maximising or minimising some objective function $l(\beta)$ with respect to β. Suppose then that the model is given by

$$\min_{\beta} / \max \; l(\beta), \tag{3.1}$$

subject to

$$\|\beta\|_p \leq t, \tag{3.2}$$

where $\|\cdot\|_p$ is the L_p-norm and the constraint in (3.2) is used to facilitate variable selection (note that the subscript p in L_p-norm has no connection to the value p used to denote the total number of variables in a design matrix). For example, when $p = 1$ in (3.2), the L_1-norm (lasso) is utilised, i.e. $\|\beta\|_1 = \sum_{l=1}^{p} |\beta_l|$. Alternatively, setting $p = 2$ in (3.2) results in the use of the L_2-norm (ridge penalty), which can be expressed as $\|\beta\|_2 = (\sum_{l=1}^{p} \beta_l^2)^{\frac{1}{2}}$. Once again, with respect to notation, a distinction should be made between the value p that is used in the summation of the regression parameters, which represents to the number of coefficients, and the subscript p in $\|\cdot\|_p$, which refers to the L_p-norm (the two are not interchangeable and have different interpretations). The value t in (3.2) is used to control the amount of shrinkage imposed on the regression parameters during model fitting. Ridge regression models tend to shrink parameter estimates towards zero, but rarely eliminates variables from the final model (predictors that exert less influence on the target often still feature in the model, but are associated with negligible coefficient estimates). While lasso also performs shrinkage on the model's coefficients, the technique has the capability of setting many regression parameter estimates exactly equal to zero. As with lasso, best subset selection requires a subset of estimated coefficients to be set exactly equal to zero, but does not shrink any of the remaining parameter estimates in the model due to the use of the L_0-norm. When applying best subset selection, the value t is set equal to q – the number of variables that user wishes to include in the final model. Lasso and ridge regression are often expressed in terms of their respective Lagrangian forms instead of the notation used in (3.1)–(3.2), which is discussed in more detail later in this chapter.

In the case of logistic regression, the objective function in (3.1) can be specified to be the log-likelihood function and subsequently maximised. Alternatively, (3.1) can be set equal to the sum of the squared errors in linear regression and minimised. The effect imposed by (3.2) on the model remains the same, regardless of the choice of $l(\beta)$. This notion applies to all variable selection procedures presented in this chapter, unless specified otherwise.

3.2 Best subset selection

Best subset regression is an extensively documented modelling approach and, due to advances in computing power and algorithmic developments over recent years, has enjoyed noticeable interest from many authors – see Bertsimas et al. (2016), Hastie et al. (2017) and Hazimeh and Mazumder (2018). The concept can perhaps be traced back to work performed by Hocking and Leslie (1967). Consider a regression problem where $\mathbf{Y}$ is a $n \times 1$ target vector and $\mathbf{X}$ is a $n \times p$ design matrix which contains the predictor variables, where n is the number of observations in the dataset. Simply put, best subset selection will proceed by fitting all possible regression models and subsequently choose the best model based on some evaluation criterion, C. Ultimately, this results in the fitting of $2^p - 1$ regression models, where each model contains less than or exactly

p predictor variables. Algorithmically, the approach can be expressed as follows:

1. For each l, $l = 1, \ldots, p$, the most optimal regression model containing l predictor variables is produced. Suppose that the best l-variable model amongst all regression models containing l inputs is denoted by $f_l(\mathbf{X}, \beta)$. Software packages such as SAS select $f_l(\mathbf{X}, \beta)$ by choosing the model with the highest likelihood ratio score or chi-square statistic in the case of logistic regression[1]. However, test statistics are not always required when evaluating each subset. The modeller can opt to use a second criterion value c, where c may or may not be the same as C, to select $f_l(\mathbf{X}, \beta)$. In their discussion on best subset selection, Kutner et al. (2005) present a variable selection example for a linear regression problem, where $\mathbf{Y} \in \mathbb{R}$, and subsequently choose the winning l-variable model that has the lowest R-square and adjusted R-square amongst all models having l inputs. Note that in the context of fitting logistic regression models by using a mixed integer linear programming approach (which is explained in more detail in Chapter 5) hypothesis tests are not considered. Instead, $f_l(\mathbf{X}, \beta)$ will be the one that delivers the largest maximum likelihood estimate.

2. Out of p possible optimal models to choose from, $f_1(\mathbf{X}, \beta), f_2(\mathbf{X}, \beta), \ldots, f_p(\mathbf{X}, \beta)$, the procedure will then conclude by selecting the regression model that results in the best obtainable value for the criterion C. For example, if C is the SSE (error sum of squares) which is often used in linear regression, the model that results in the lowest SSE will be put forth as the final regression model.

The criterion C is in most cases a subject of choice specified by the modeller. Ultimately, the choice of C will determine the regression model that is selected as the final model. Measures that may be commonly used in best subset selection include the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC). Acquah (2010) performed a full comparison between the AIC and BIC and provides a detailed explanation of both, which follows next. The Akaike Information Criterion attempts to select a model that maximises the likelihood of the regression function, but simultaneously penalises the model for including too many parameters (Akaike, 1973). The criterion is given by

$$AIC = -2\log(L) + 2q \tag{3.3}$$

for logistic regression and

$$AIC = n\log\left(\frac{SSE}{n}\right) + 2q \tag{3.4}$$

for linear regression.

[1]See Appendix A for a detailed explanation of the likelihood ratio test.

The Bayesian Information Criterion (also known as the Schwarz Bayesian Criterion or SBC) is similar to the AIC and penalises the model for the inclusion of large numbers of predictors (Schwartz, 1978). For logistic regression, the criterion is expressed as

$$BIC = -2\log(L) + q\log(n), \tag{3.5}$$

whereas linear regression makes use of

$$BIC = n\log\left(\frac{SSE}{n}\right) + q\log(n). \tag{3.6}$$

For both the AIC and BIC, the value q represents the number of fitted parameters. Lower AIC or BIC values are assumed to be associated with more superior models, which means that model selection techniques such as best subset selection will favour increasingly negative criterion values. Notice that the first half of both measures are the same, with only the second term that differs. Specifically, the second term of the BIC is also dependent on the size of the data. Due to the inclusion of n in the BIC penalty, the BIC tends to select more parsimonious models for larger datasets and is therefore often favoured as a more robust measure over the AIC (Kutner et al., 2005). However, many authors within the modelling community suggest using both the AIC and BIC in conjunction with one another to arrive at a final decision.

Whether the dependent variable is continuous or binary may exert more influence on the choice of certain measures, as opposed to the AIC and BIC, which can be used comfortably for both linear and logistic regression models. In linear regression, for example, popular and well-known criteria such as the SSE or R-squared are often utilised. The SSE is a simple calculation and is interpreted as the sum of the squared differences between the actual and predicted values, which is given by $\sum_i (Y_i - f(\mathbf{X}, \beta))^2$. The R-squared, also known as the coefficient of determination, is calculated as

$$R^2 = 1 - \frac{SSE}{SSTO}, \tag{3.7}$$

where the SSTO is known as the *total sums of squares* and is expressed as $SSTO = \sum_i (Y_i - \bar{Y})^2$. The value $\bar{Y}$ represents the mean of the dependent variable vector $\mathbf{Y}$. It should be noted that the R-squared criterion is bounded between zero and one, or $0 \leq R^2 \leq 1$.

Models that produce a lower SSE are generally favoured. Notice from equation (3.7) that lower SSE values will result in a higher R-squared value. Therefore, regression lines based on subsets with a higher R-squared are often chosen as superior models. The SSE and R-squared criterion will frequently produce similar or identical final models, since both aim to minimise the difference between the values that have been observed and those that have been estimated. For this reason, these two measures are regularly used interchangeably.

In the case of logistic regression, where binary dependent variables are being dealt with, evaluation metrics that are typically assessed include the misclassification rate, accuracy, AUC (area under the curve) or Gini statistic. A sufficient explanation of these measures is provided by Patetta (2012) which is discussed next.

First, let TP and TN denote the total number of true positives and true negatives in a binary prediction problem, where $Y_i \in \{0, 1\}$. If the i-th observation has experienced the event, or $Y_i = 1$, and the model has also made the correct prediction, $\hat{Y}_i = 1$, then the i-th observation is considered to be a single true positive. Alternatively, if observation i is associated with a non-event, $Y_i = 0$, and has been classified correctly by the model such that $\hat{Y}_i = 0$, then it will be seen as a true negative. Given this notation, the accuracy of the model is defined as

$$Accuracy = \frac{TP + TN}{n}. \tag{3.8}$$

In (3.8), TP is a count of *all* true positives, whilst TN counts *all* of the true negatives. As before, n is the total number of cases. It should be intuitive that a model with a higher level of accuracy is generally favoured and that the accuracy statistic can neither be less than zero nor be more than one.

Using the same notation, let FP represent the total number of false positives and FN the total number of false negatives. A false positive can be seen as the converse of a true positive and is defined as an observation where the actual value is a non-event, or $Y_i = 0$, but the model has predicted otherwise, i.e. $\hat{Y}_i = 1$. Similarly, a false negative refers to a case that did indeed experience the event, $Y_i = 1$, but was classified as a non-event, $\hat{Y}_i = 0$. The misclassification rate can then be calculated as

$$Missclass = \frac{FP + FN}{n}. \tag{3.9}$$

Unlike the accuracy of the model, a lower misclassification rate is preferred.

The area under the curve and Gini statistic (Schechtman and Schechtman, 2016) are two very similar measures with analogous interpretations. Specifically, the one can be derived by using the other. Both relate to the Receiver Operator Characteristic curve or ROC curve in binary models. The ROC curve encapsulates the ability of the model to isolate "good" observations (those with $Y_i = 0$) earlier on where the posterior probabilities of the model are low, while "bad" cases (with $Y_i = 1$) are "captured" at higher probabilities [2]. Ultimately, Gini and AUC measures can be seen as criteria used to assess a binary model's ability to rank its probabilities well. Refer to Appendix B for a better understanding of the ROC curve.

Let π_{co} be an arbitrary probability cut-off, where $CG_{\pi_{co}}$ is the cumulative proportion of

[2]The notion of a case with $Y_i = 0$ being referred to as a "good" and an observation with $Y_i = 1$ being known as a "bad" originated from credit scoring, where a customer who defaulted on a credit product over a certain period of time was seen as an event and assigned a value of one, whereas those who remained up to date over the period of the loan were seen as non-events and assigned a value of zero.

goods that have a predicted probability of π_{co} or less. Similarly, let $CB_{\pi_{co}}$ be the cumulative proportion of bads at π_{co} with predicted probabilities $\pi_i \leq \pi_{co}$. Suppose that a grid consisting of Π cut-off probabilities is considered, where $0 \leq \pi_{co}^j \leq 1$ for $j = 1, \ldots, \Pi$ with $\pi_{co}^j < \pi_{co}^{j+1}$ and $\Pi \in \mathbb{Z}^+$. The Gini statistic is then given by

$$Gini = \sum_{j=1}^{\Pi-1}[(1-CG_{\pi_{co}^j})-(1-CG_{\pi_{co}^{j+1}})] \times [(1-CG_{\pi_{co}^j})+(1-CG_{\pi_{co}^{j+1}})-(1-CB_{\pi_{co}^j})-(1-CB_{\pi_{co}^{j+1}})].$$

The AUC can then be calculated as

$$AUC = \frac{Gini + 1}{2}. \tag{3.10}$$

The choice of using the Gini statistic or AUC is a subjective one and ultimately depends on the modeller's preference, since both measures will most likely put forth the same final model. Within the financial services industry (especially banking), the Gini is often used as the measure of choice.

For both the AUC and Gini, higher values are associated with better models. In general, models that have high AUC and Gini statistics tend to display improved levels of accuracy and lower misclassification rates. Therefore, it should be noted that the majority of the measures introduced for logistic regression will most likely point to the same subset of models as the most superior models in best subset selection.

While subset selection is a computationally expensive variable selection technique to consider (which will be discussed shortly), certain approaches have been proposed to reduce the algorithmic burden of the procedure. Firstly, the modeller can determine the best q-variable model, with $q \leq p$, where improvement in the measure C starts to diminish. In other words, fitting a model with more than q variables does not significantly improve the quality of the model, meaning that any regression model containing $q + 1, \ldots, p$ inputs results in a negligible incremental increase or decrease in C. By determining the index q at which C flattens out, two concerns in best subset selection are addressed. Firstly, it is no longer necessary for the procedure to consider all $2^p - 1$ possible models. Instead, only those containing q or less variables need to be evaluated. Secondly, this approach prevents the model from overfitting [3].

In general, the continuous introduction of more variables into the regression equation increases the variance of the model, but simultaneously significantly reduces its bias. This often leads to a steady decrease in the SSE in linear regression or constant increase in the maximum likelihood estimate in logistic regression. This is usually the result of the model finding spurious

[3]Overfitting refers to an instance where models are trained to predict the outcome of the development dataset as perfectly as possible, which often produces a model that does not generalise well and displays poor performance on new data, such as the test dataset or samples taken from a different time period.

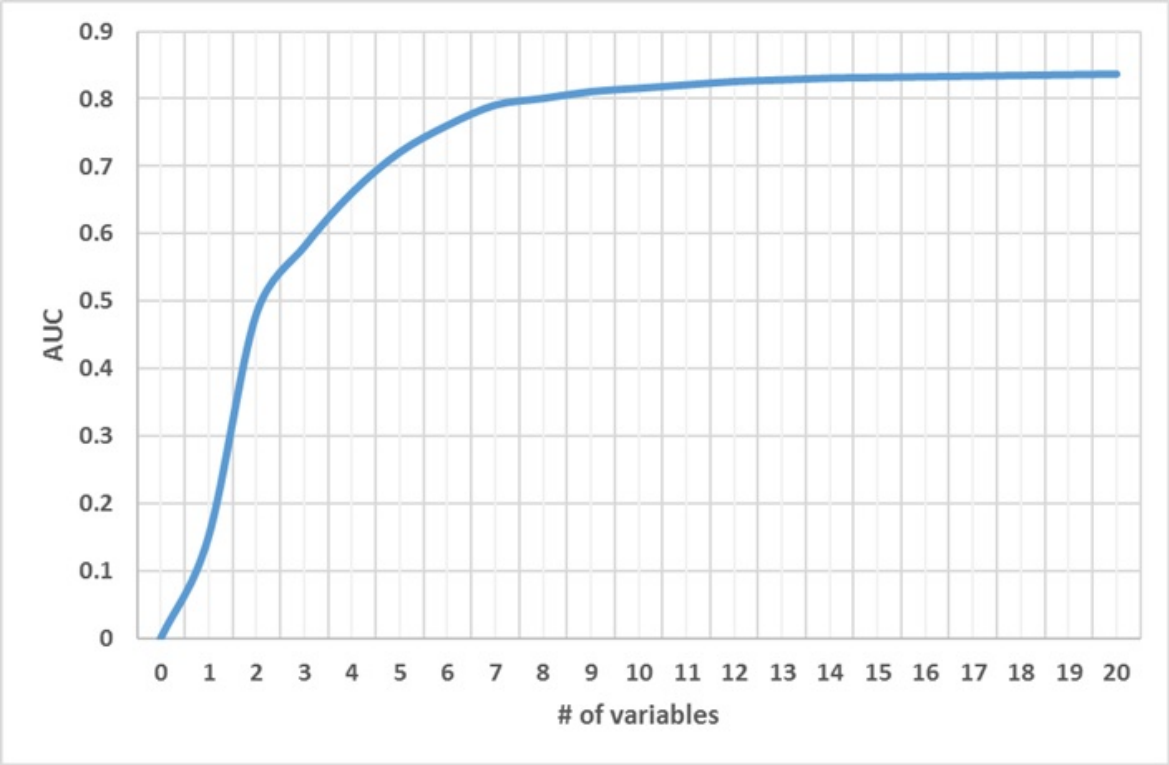


Figure 3.1: Example of diminishing improvement in AUC

(and in many cases, nonmeaningful) correlations between the dependent variable and the predictors. Ultimately, this implies that most measures that have been proposed in this section as suitable candidates for C will continuously improve as more and more predictors are added to the model. Empirical studies have shown that even criteria such as the AIC and BIC also have a tendency of favouring models that contain an overabundance of variables, unless n is noticeably large (Kutner et al., 2005). Clearly, such a result defeats the purpose of variable selection.

Finding the point $q \leq p$ at which the improvement in C tapers off serves as a suitable solution to the aforementioned drawback. To illustrate the idea, consider a logistic regression problem with $p = 20$ potential predictors. The AUC for the best l-variable model, where $l = 1, \ldots, 20$, is plotted in Figure 3.1. Notice that the AUC increases rapidly at first as more inputs are added to model. However, at some index q, $1 \leq q \leq 20$, a point of saturation is achieved where the AUC levels off and cannot be significantly improved by increasing the number of predictors in the model. Figure 3.1 shows that a model containing roughly 7 to 9 variables is sufficient, after which no real improvement in model fit is realised. The modeller can therefore opt to only consider regression models containing $q \leq 9$ variables in the best subset procedure.

In general, the idea of considering only q out of p predictors for best subset regression is discussed by Miller (2002) and studied by Bertsimas et al. (2016) and Hastie et al. (2017). Given a response vector $\mathbf{Y} \in \mathbb{R}$ and a design matrix $\mathbf{X}$ with dimensionality p (excluding the

intercept term), Miller (2002) proposes that the best subset selection problem finds the subset of q predictors, with $0 \leq q \leq \min(n, p)$, that results in the best model fit by solving the following objective function for linear regression:

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\beta\|_2^2, \tag{3.11}$$

subject to

$$\|\beta\|_0 \leq q, \tag{3.12}$$

where $\|\cdot\|_0$ is the L_0-norm and counts the number of non-zeros in β, or $\|\beta\|_0 = \sum_{l=1}^{p} I(\beta_l \neq 0)$ and $I(\cdot)$ is the indicator function.

The authors note that, given a total of p potential predictors, the modeller may wish to consider only q of these for inclusion in regression. For example, if $n = 50$ and $p = 100$, a suitable model may be found by relying on only $q = 10$ inputs. In banking, governance structures for credit scoring models stipulate that analysts are allowed to include at most 15 to 20 explanatory variables in a scorecard model. The aforementioned limitation is imposed in support of improved understanding of credit models and ease of implementation. Specifically, by limiting the problem to a subset of q predictors, the best subset procedure is only required to assess $\binom{p}{q}$ models. This alleviates some of the computational burden imposed by the procedure, since $\binom{p}{q} < 2^p - 1$.

Finally, it should be noted that best subset selection is not a polynomial time procedure. As stated by Miller (2002), each additional variable that is added for consideration doubles the execution time of model selection. Ultimately, this means that subset selection approaches are exponential-time procedures. Most authors agree that traditional methods for implementing best subset selection become computationally infeasible when the dimensionality of the problem exceeds roughly 40 to 60 potential predictors. Potts and Patetta (1999) explain that the approach is an impractical variable selection method when considering data sets comprising of more than approximately 50 inputs. Lund (2017) argues that analysts who make use of SAS should consider abandoning best subset selection when the number of variables is 75 or more, citing a rapid increase in execution time. Specifically, best subset selection is considered to be an NP-hard problem (Natarajan, 1995). Software packages like SAS use branch-and-bound algorithms to obtain the best q-variable models. Branch-and-bound algorithms and the definition of NP-problems are discussed in more detail in Chapter 4. Additionally, a thorough discussion on a popular branch-and-bound method used for performing best subset selection is provided in Appendix C.

3.3 Stepwise selection

Forward selection, backward elimination and stepwise selection represent the three main components of what is commonly known as stepwise regression methods – a set of variable selection techniques that aim to be computationally friendly alternatives to the best subset selection regression problem. While best subset selection procedures attempt to find the best model by estimating several regression models, stepwise methods aim at producing a "good" model by fitting a single regression model that approximates the relationship between the target variable and a subset of predictors.

In order to explain the inner workings of stepwise selection procedures as set out in Kutner et al. (2005), the hypothesis tests utilised by these methods are first introduced. In logistic regression, let the matrix of second-order partial derivatives, or Hessian matrix, of the log-likelihood function with respect to the regression coefficients be denoted by $\mathbf{H}$. The ij-th entry of $\mathbf{H}$ is given by

$$\frac{\partial^2 \log L(\beta)}{\partial \beta_i \partial \beta_j}.$$

Suppose that $\hat{\beta}$ is the vector of maximum likelihood estimates obtained by the logistic regression model for β. If $\mathbf{H}$ is evaluated at $\beta = \hat{\beta}$, then the estimated variance-covariance matrix of the regression coefficients can be expressed as

$$s^2(\hat{\beta}) = \left([-\mathbf{H}]_{\beta=\hat{\beta}} \right)^{-1}.$$

If the sample size n is sufficiently large enough and the regression problem contains p variables (excluding the intercept term), then the statistic given by

$$\frac{\hat{\beta}_l - \beta_l}{s(\hat{\beta}_l)} \text{ for } l = 1, \ldots, p$$

follows a standard normal distribution and $s(\hat{\beta}_l)$ serves as an approximated standard deviation for $\hat{\beta}_l$.

The two-sided Wald test for variable selection then tests the following hypothesis:

$$H_0 : \beta_l = 0$$

vs

$$H_a : \beta_l \neq 0$$

The test statistic is calculated as

$$Z^* = \frac{\hat{\beta}_l}{s(\hat{\beta}_l)}, \tag{3.13}$$

where $Z(\cdot)$ is the standard normal distribution evaluated at $1 - \frac{\alpha}{2}$ and α is a prespecified confidence level.

The decision to reject H_0 is made if $|Z^*| > Z(1-\frac{\alpha}{2})$. The square of Z^* is also regularly used, resulting in a test statistic that follows a Chi-square distribution with one degree of freedom, i.e. $(Z^*)^2 \sim \chi^2_1$. This is known as the Wald Chi-square test. The decision to reject H_0 is made if $(Z^*)^2 > \chi^2(1-\alpha, 1)$, where $\chi^2(1-\alpha, df)$ is the $(1-\alpha)100$ percentile of the Chi-square distribution with df degrees of freedom. The p-value associated with the test statistic is given by the right-tailed probability of the χ^2_1-distribution evaluated at $(Z^*)^2$, denoted by $P(X > (Z^*)^2)$ with $X \sim \chi^2_1$. Following the aforementioned notation, H_a is therefore accepted when the p-value is less than α. In practice, the significance level α is usually set equal to 0.01, 0.05 or 0.1 and is sometimes referred to as the p-value cut-off. When performing variable selection, SAS makes use of the Chi-square test statistic (Patetta et al., 2009).

The forward selection procedure is initiated by starting with an empty model, i.e. a regression line that contains no predictors. At each step, the Chi-square statistic of every variable that is not included in the current model, is computed. Logically, the test statistic for all predictors will be calculated during the first iteration. The largest of these test statistics is then examined and the variable associated with it is added to the model if the statistic is significant at the specified confidence level α. Once a variable has been selected for inclusion, it may never exit the model. The aforementioned process is repeated until none of the remaining variables exhibit test statistics that are significant at α.

As an alternative to forward selection, backward elimination starts with a full model that comprises of all of the potential inputs at its disposal. The Wald Chi-square test is then performed for every variable that is currently included in the model. As with forward selection, this implies that a test statistic is calculated for every one of the p predictors at the first step. During each iteration, the variable with the least significant test statistic value at the confidence level α (smallest Chi-square statistic) is selected for removal from the model. Once a variable is eliminated, it may not re-enter the model at any point going forward. The backward elimination procedure is repeated until the smallest test statistic obtained is statistically significant for inclusion in the model, i.e. H_a for (3.13) is concluded for the lowest Wald Chi-square measure.

Stepwise selection may be viewed as an amalgamation of forward and backward regression. Similar to forward regression, the procedure starts with an empty model and adds one variable at a time. Once more, the Wald Chi-square statistic is calculated for each input and the variable associated with the most significant test statistic is selected for inclusion. However, unlike forward selection, the algorithm does not proceed to the next step from here. At this point,

variables that are currently found in the regression do not necessarily remain until the next iteration. The backwards component of the procedure is now initiated and may opt to remove a variable from the model produced in the most recent step by nominating the predictor with the lowest test statistic that is not significant at α. The stepwise procedure will iteratively progress in such a manner until none of the remaining variables meet the criteria for inclusion in the model (as with forward selection) or if the most recent variable that was added/removed in the current step is the same variable that was removed/added during the previous iteration.

It should be noted that the choice of significance level α will exert an influence on the variables chosen by stepwise procedures. Choosing α to be strict, such as $\alpha = 0.01$, will yield more parsimonious models containing less features. However, the modeller runs the risk of excluding potentially predictive inputs. Alternatively, relaxing the criteria by setting $\alpha = 0.1$ may result in a larger subset of variables being included in the final model. In such instances, the model might opt to select weaker or more noisy inputs for inclusion. In most software packages, the default value for α is set at 0.05, unless specified otherwise.

Stepwise procedures are notably one of the most popular variable selection techniques used when fitting regression models in practice. Whittingham et al. (2006) show that a study of articles published in 2004 in three reputable ecological and behavioural journals revealed that stepwise selection was the method of choice in 57% of papers which involved a multiple regression approach. After considering the publications in two leading Chinese epidemiology journals between 2004 and 2008, Liao (2010) found that 44% opted to use stepwise procedures when regression analyses were applied. In banking, stepwise selection is arguably the most common (and often only) variable selection technique used when fitting logistic regression models during scorecard development. In fact, when credit scoring models are presented to banking technical committees, analysts who prefer to explore alternative methods may be required to motivate why they *did not* make use of stepwise procedures. However, in spite of their popularity, many authors have made extensive arguments against the use of these methods, which will be discussed next.

Firstly, stepwise variable selection methods are widely criticised for their reliance on p-values in selecting inputs. Mantel (1970) states that tests aimed at performing comparisons are only reliable if prespecified models are compared. However, tests conducted during stepwise procedures are not based on prespecified models, as the models being compared are the result of several previous steps that included or deleted some of the variables. Consequently, multiple testing will generally lead to underestimated p-values. Additionally, the p-values obtained from these tests do not necessarily determine whether a single variable is relevant, but rather whether the particular variable is relevant given the specific set of predictors already included in the model (which may not provide a true reflection of the variable's predictiveness). Patetta et al. (2009) present a similar argument to Mantel (1970) by noting that constantly refitting the model

during various iterations distorts the significance levels of conventional hypothesis tests, where standard errors, test statistics and p-values are calculated as if the entire model was prespecified to begin with. This may result in a scenario where the importance of certain predictors are overstated, leading to overly optimistic predictions. Since p-values are underestimated, traditional significance levels like $\alpha = 0.05$ may yield biased inferences.

Smith (2018) references the works of several authors who argue that single test statistics are not appropriate when used in a sequence of steps. Incorrect p-values may lead to confidence intervals for parameter values which are too narrow and can potentially result in the model being overfitted to the training data. Additionally, the author also mentions that the p-values are more subjected to chance as the number of potential variables increases. For example, if a modeller is faced with a regression problem comprising of $p > 30$ inputs, it is highly likely that at least one (or more than one) variable will be statistically significant at α, even if all p effects are simply nuisance variables with no true relationship with the response at all. After presenting many empirical results, Constanza and Afifi (1979) indicate that the reliance on conventional significance levels, such as $\alpha \in \{0.01, 0.05, 0.1\}$, may yield misleading results, unless the modeller is confident that only a specific few variables will be meaningful additions to the model (something which is rarely known beforehand in real-world applications). Ultimately, traditional choices for α may cause the model to overlook some important features.

Adding to the results provided by the aforementioned authors, Kutner et al. (2005) discuss simulation studies which have shown that relaxing the confidence level α results in a weak model when many of the predictors are found to be uncorrelated with the response.

Secondly, the profound influence that the size and dimensionality of the modelling set exerts on the results obtained from stepwise procedures have been well documented in literature. Hosmer and Lemeshow (2000), Derksen and Keselman (1992) and Potts and Patetta (1999) stipulate that large significance levels might be more appropriate for smaller datasets, whereas smaller p-value cut-offs are better suited for larger datasets. The authors argue that a blanket approach cannot be universally applied to all regression problems when selecting significance levels for variable inclusion or deletion, as the predictors selected by the final model will depend quite drastically on the size of the sample and not necessarily on what is in the dataset. Patetta et al. (2009) provide an excellent example in the form of a table that demonstrates how different p-values are interpreted as either weak or strong for a given sample size, where p-values that are seen as robust for smaller sets are shown to be unreliable for larger ones. In their simulation studies conducted on stepwise selection, Bursac et al. (2008) show how dependent stepwise selection is on the number of cases in the modelling set. The authors generate simulated data wherein the true underlying regression parameters of certain variables were specified beforehand, whereas the rest of the parameter values were set equal to zero. This meant that the true model was known in advance. In some simulations, the authors also introduced confounding effects into

the model, where the distribution of one variable depends on another variable. The percentage of times where stepwise procedures chose the correct model was then studied. Results indicated that the accuracy of the models differed significantly for sufficient increases or decreases in the sample size. Specifically, it was mentioned that backward elimination and stepwise selection performed noticeably worse for smaller datasets as opposed to forward selection, which fared slightly better. Alternatively, forward selection appeared to be substantially worse off compared to its peers when larger training sets were introduced.

In order to address the problem of biased inferences made on p-values, Patetta et al. (2009) suggest splitting the dataset such that the model is fit on one portion and inferences are made on the other. However, the performance of the model and its parameters will once more be affected by the size of the data. Furthermore, it is noted that such an endeavour is nearly impossible for smaller sample sets.

When the predictors of the model are considered, several pitfalls have also been noted. Stepwise selection is seen as an inefficient variable selection technique coupled with deteriorating performance when the number of inputs exceeds roughly 60 variables (Potts and Patetta, 1999), whereas backward selection becomes nearly impossible when faced with a large subset of effects (Smith, 2018). Derksen and Keselman (1992) mention that the degree of multicollinearity[4] present in the data affects the frequency with which variables enter and exit the model. The authors also state that the number of predictive inputs which enter the model is influenced by the number of noisy variables that are considered by the stepwise procedure. Lastly, Harrel (1997) explains that models with a sparse data structure result in unstable parameter estimates which should be treated as suspect.

Another concern to be raised is the subjectiveness inherent to stepwise procedures and their heuristic nature. Specifically, the choice of p-value cut-offs and which stepwise procedure to utilise, both of which need to be specified by the user, will determine the final model that is obtained. It has already been mentioned previously that significance level choices will influence the number of variables chosen by the model, where less features are considered for inclusion for smaller values of α. In addition, the different procedures tend to yield dissimilar results. Backward elimination performs better than its counterparts when a large degree of multicollinearity is detected in the data. The technique is also preferred when the dimensionality of the problem p is relatively small, since the variances of the estimated parameters are inflated during the initial steps of forward selection due to variables that have not yet entered the model. For backward elimination, variances tend to be more unbiased as a result of important predictors that are retained at each step. Alternatively, forward regression may be superior in the presence of strong interactions that exist between predictors. In stepwise selection, when the criterion

[4]The term multicollinearity refers to the existence of significant correlations between input variables, otherwise known as inter-variable correlations.

for entering the model is different from the p-value cut-off at which a variable exists the model, the repeated cycling of predictors may occur. The subjective nature of these procedures means that the final solution depends heavily on the choices made by the modeller and not necessarily on the relationship that exists between the predictors and the response.

Ultimately, the modeller is rarely guaranteed an optimal solution with stepwise selection methods. Kutner et al. (2005) highlights the fact that forward, backward and stepwise procedures regularly deliver sub-optimal regressions as final models. Since these techniques only end up with a single model, it remains unknown whether or not a potentially better model exists – perhaps one that is even more parsimonious than the one given by stepwise selection. Alternatively, best subset selection explores every possible avenue. Thompson (1995) reasons that stepwise selection procedures will mainly produce only locally optimal models, stating that the l-th variable added to a model that already contains $l-1$ inputs does not necessarily result in the best l-variable model, but rather that the procedure bases its choice on the nature of the model that was obtained in the previous iteration. Therefore, an exact, optimal solution is never guaranteed by executing a once-off fit to the data using these techniques. Many of the authors mentioned in this section argue that stepwise selection methods should not be seen (as they often are) as the holy grail of finding the best subset of variables. Instead, they should be considered as exploratory tools that suggest potentially viable models, the nature of which should be thoroughly investigated. Analysts are also motivated to repeatedly fit regression models using various incarnations of these procedures and subsequently use subject matter expertise to make a final decision.

3.4 Variable selection via regularisation

Penalised regression methods, such as lasso, ridge regression, the smoothly clipped absolute deviation penalty (SCAD), minimax concave penalty (MCP) and the elastic net have also been proposed in an attempt to combat the computational burden imposed by subset selection. These methods have the added advantage of shrinking regression coefficients towards zero while simultaneously setting many coefficients exactly equal to zero (however, ridge regression is notorious for not producing a sparse estimated parameter vector). All of the methods explained in this section are based on the premise of penalising the model for selecting too many variables for inclusion by placing constraints on the regression coefficients during model fitting.

3.4.1 Ridge regression and the lasso

Hoerl and Kennard (1970) motivate the use of $[\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_p]$, $\lambda \geq 0$, instead of $\mathbf{X}^T\mathbf{X}$ when estimating the parameter vector β in regression analysis, noting that certain errors may arise if

$\mathbf{Y} = \mathbf{X}^T\beta$ is specified as the true model and $\mathbf{X}^T\mathbf{X}$ is not a unit matrix. In such cases where $\mathbf{X}^T\mathbf{X}$ is not invertible, least squares estimates do not make sense within the realm of physics, engineering and chemistry. Additionally, cases where the β's are unconstrained may lead to models that are susceptible to high variance. Potential alternatives to these problems often require the analyst to treat the model as a "black box" or to eliminate certain factors from the model, thereby eradicating correlation bonds that exist amongst the predictors used to form $\mathbf{X}^T\mathbf{X}$. The authors view both alternatives as unsatisfactory and unwanted, which serves as motivation for their work. The first section of the authors' work provides a thorough description of the assumptions made in traditional least squares estimation, the properties of β and how these inherent characteristics of least sqaures estimation might lead to the aforementioned problems. In an attempt to combat the shortcomings of ordinary least squares (OLS), Hoerl (1964) suggested the use of the following matrix when estimating β:

$$\beta^{ridge} = \left[\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_p\right]^{-1}\mathbf{X}\mathbf{Y}, \quad \lambda \geq 0,$$

where $\mathbf{X}$ is standardised and $\mathbf{Y}$ is centered. The inclusion of λ makes the model non-singular, even if $\mathbf{X}^T\mathbf{X}$ is not invertible. This leads to the following least squares problem:

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2, \tag{3.14}$$

subject to

$$\sum_{l=1}^{p} \beta_l^2 \leq t, \tag{3.15}$$

where (3.15) is known as the *ridge constraint* . Taking the Lagrangian form of this equation produces the penalised model known as the *ridge regression model*:

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_2^2, \tag{3.16}$$

where $\|\cdot\|_2$ is the L_2-norm. Notice that for the Lagrangian form of (3.16) in logistic regression, the first half of the equation containing $\|\mathbf{Y} - \mathbf{X}\beta\|_2^2$ can be replaced with $-\log L(\beta)$.

The model in (3.16) might yield a lower prediction error than a straightforward OLS fit due to a lower degree of variance. This is achieved by shrinking the coefficients, thereby increasing the bias. Note that t in (3.15) and λ in (3.16) control the amount of shrinkage and are subsequently known as the shrinkage parameters.

In order to choose an appropriate value for λ, Hoerl and Kennard (1970) suggest a graphical approach dubbed as the ridge trace. In such a graphical representation, the estimated regression coefficients are plotted as a function of λ. It is then up to the modeller to select a value for λ at a point where the estimates appear to have stabilised and have sensible signs. It is perceived that

the coefficients vary wildly for smaller values of λ, after which they tend to stabilise. Therefore, a possible option would be to select the smallest value of λ at which most estimates start to remain constant throughout the trace (note that a very large λ will introduce high levels of bias into the model, ergo the smallest possible value is recommended). However, this method for selecting λ has been criticized by many due to the fact that it has no objective basis. Alternatively, another popular method for choosing λ in (3.16) or t in (3.14)–(3.15) is by parameter tuning conducted via k-fold cross-validation. Cross-validation is a technique used to evaluate predictive models by partitioning the original sample into a development set for training the model and a test set to evaluate it. Specifically, in k-fold cross-validation, the original sample is randomly partitioned into k equally sized subsamples. Of the k subsamples, a single subsample is utilised for testing the model, while the remaining $k-1$ portions are used for training the model. The cross-validation process is repeated k times wherein each one of the smaller samples is used exactly once to validate the model. Finally, the average of the k results can then be used to obtain a consolidated measure. In the cross-validation exercise, ridge regression is executed for a grid of λ entries, where the λ value that results in the lowest combined prediction error is selected. Utilising k-fold cross-validation means that *all* of the data is used for both training and testing the model, which is an added advantage.

Ridge regression is a continuous process that shrinks its coefficients in order to obtain more stable models. However, none of the coefficients are set exactly equal to zero, which results in a model that is not easily interpretable (Tibshirani, 1996). By never setting any of the estimated parameters equal to zero, ridge regression fails at producing sparse models and is subsequently regarded as a less successful variable selection technique.

As an alternative, Tibshirani (1996) proposes the *lasso* (least absolute shrinkage and selection operator), which attempts to overcome the high level of variance inherent to ridgre regression models (due to the inclusion of many non-zero parameters) by setting many regression coefficients exactly equal to zero. The end result is a more interpretable and parsimonious model. As with ridge regression, the lasso also introduces bias into the model by shrinking its coefficients. Emperical studies have shown that lasso enjoys many favourable properties associated with both best subset selection and ridge regression. By only selecting a limited number of estimates to be greater than zero, the lasso is able to yield sparse regression coefficient vectors like those seen in best subset selection. Additionally, lasso tends to deliver stable models that generalise well on new sets of data due to its biased estimates.

Breiman (1993) introduced the nonnegative garrotte, which serves as motivation for the lasso and can be expressed as

$$\min_{\beta} (\mathbf{Y} - \mathbf{c}\beta\mathbf{X})^2, \tag{3.17}$$

subject to

$$\sum_{l=1}^{p} c_l \leq t, \quad c_l \geq 0. \tag{3.18}$$

The author shows that the model in (3.17)–(3.18) consistently produces lower prediction errors when compared to other selection techniques and is a worthy opponent for ridge regression, except when the true underlying model has many small non-zero coefficients. However, a major drawback of (3.17)–(3.18) is that the solution depends on the magnitude and sign of the estimates. In highly correlated settings, the model may perform especially poor. Lasso, on the other hand, avoids the explicit use of regression estimates.

Lasso regression models bear a striking resemblance to the ridge regression model given in (3.14)–(3.15) and (3.16). Using the same notation as before, the lasso estimate can written as

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2, \tag{3.19}$$

subject to

$$\sum_{l=1}^{p} |\beta_l| \leq t, \tag{3.20}$$

or in Lagrangian form

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_1, \tag{3.21}$$

where $\|\cdot\|_1$ is the L_1-norm, i.e. $\|\beta\|_1 = \sum_{l=1}^{p} |\beta_l|$. Once again, the part of (3.21) containing $\|\mathbf{Y} - \mathbf{X}\beta\|_2^2$ can be replaced with $-\log L(\beta)$ for the Lagrangian form in logistic regression modelling.

Figure 3.2 shows graphically why lasso produces a sparse coefficient vector and ridge regression does not (Tibshirani, 1996). The objective function in (3.14) and (3.19) is represented by the elliptical contours whereas the constraint regions in (3.15) and (3.20) are given by the greyed-out areas. In both cases, a solution is obtained at the point where the contours touch the grey constraint region, which is indicated by the blacked out coordinate. For lasso, this happens at a corner of the constraint region (in the two-dimensional case, the square), which corresponds to one or more estimated coefficients being zero. There are no corners that exist for ridge regression, which implies that solutions which are exactly equal to zero will rarely be found. Ultimately, Figure 3.2 highlights the benefit of choosing the L_1-norm over of the L_2-norm in the pursuit of parsimonious models and therefore the competitive advantage that lasso has over ridge regression.

As with ridge regression, cross-validtion is often employed to fit lasso models. Tibshirani (1996) explains that the lasso is executed for an index parameter $s = (t/\sum \hat{\beta}_l)$, where the prediction error of the model is obtained over a grid of s-values ranging between 0 and 1 inclusive. The model at the value of s yielding the lowest error is then selected. In addition, the authors

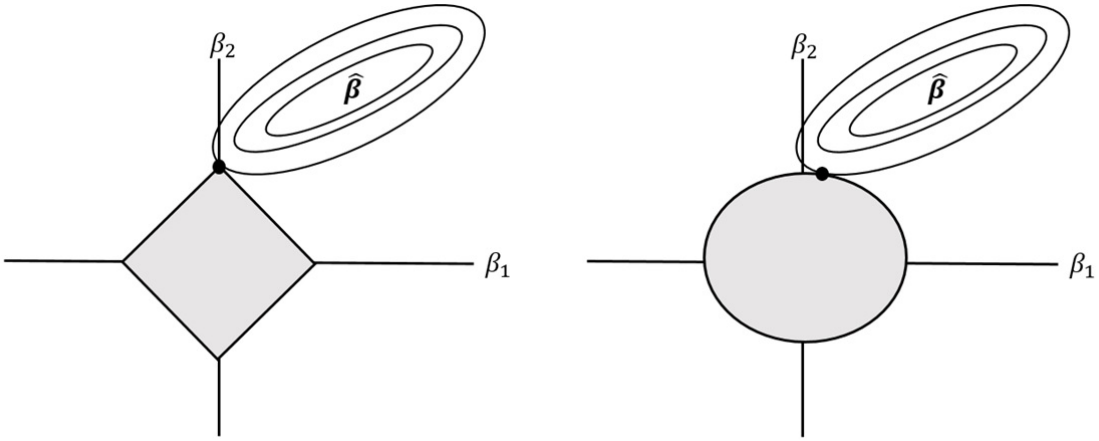


Figure 3.2: Graphical example of lasso vs ridge regression

also present two alternative methods for estimating the shrinkage paratmeter in lasso.

Lasso and ridge regression are associated with many advantageous characteristics. Both are computationally friendly convex optimisation problems that are easy to implement and can be solved efficiently by various commercially available solvers (Nesterov (2007); Efron et al. (2004); Friedman et al. (2007)). For example, Efron et al. (2004) proposed the use of a least angle regression or *LARS* algorithm, which is very effective in computing an entire path of lasso solutions in the same order of time as a single OLS model fit. Furthermore, models obtained from these approaches are generally stable and unaffected by small changes in the data. Increased stability is achieved by forgoing the need to obtain unbiased parameters, thereby producing estimates with less variance (Neter et al. (1983); Rawlings (1988); Myers (1990)). Ridge regression models seem to generalise especially well to new data where the values of the variables fall outside the region of observations on which the model is trained (Neter et al., 1983). Alternatively, ordinary least squares methods may perform very poorly in such cases. Ridge regression is also considered to be particularly valuable when it is known or believed beforehand that the coefficient estimates associated with the predictors cannot be very large (Draper and Smith, 1998). On the other hand, it has been consistently proven that lasso delivers sparse models with good predictive performance (Buhlmann and van-de Geer (2011); Hastie et al. (2009)). For a small to moderate number of moderately size features, lasso does exceptionally well in producing parsimonious models and may even outperform best subset selection by a significant margin.

However, in spite of many beneficial properties, lasso and ridge regression approaches are not without drawbacks. Firstly, the choice of shrinkage parameter λ remains a judgemental one, meaning that these models still suffer from some degree of subjectiveness, even when λ is chosen by the use of data-driven methods such as cross-validation (Neter et al., 1983). Increasing the shrinkage parameter in the pursuit of a sparse model causes lasso to set too many estimated

coefficients equal to zero and ridge regression to severly shrink its estimates, which may drastically reduce or completely eliminate the influence of potentially important predictors on the response variable (Bertsimas et al., 2016). Alternatively, a shrinkage parameter that is too small might result in the inclusion of too many noisy variables in the final model. Notice that the problematic subjectiveness inherent to the choice of λ in lasso and ridge regression is similar to the problem of choosing the correct p-values in stepwise regression methods.

For problems where no prior knowledge exists on how big the true underlying regression coefficients possibly are or where upper and lower bounds do not apply on the regression parameters (the β's are unrestricted), lasso and ridge regression will produce completely inappropriate models (Draper and Smith, 1998). Additionally, when multicollinearity is severe, ridge regression will attempt to reduce inter-variable dependencies by aggresively shrinking some of the coefficients. However, the variables will still remain active in the model. According to Ryan (1997), the user should rather opt to drop one or more inputs from the training set in such circumstances, especially when many inputs are almost perfectly correlated. In the presence of many noisy or highly correlated inputs, lasso tends to include a large number of nuisance variables in the final model (all with coefficients shrunk towards zero) in an attempt to find a model with superior predictive performance (Bertsimas et al., 2016).

Heinze et al. (2018) explain that these models often yield estimates which are difficult to interpret in the context of explanatory and descriptive models, due to the bias inherent to the parameters. Lasso and ridge regression can therefore potentially hinder inferences made in medical research domains, where the interpretation of the effects in a model is key. The problem of performing valid inference on estimated regression coefficients in lasso have been investigated by authors such as Taylor and Tibshirani (2015) and potential solutions have been proposed. However, there still remains a lack of evidence on the performance of the remedial measures suggested by the authors. Ultimately, in order to perform effective inference, the user requires that both the bias and variance component of regression coefficients are captured. This dictates the need for an estimate of bias, which cannot be obtained from variance-reducing methods such as lasso and ridge regression. Alternatively, when variables are included in a model via best subset selection, it is done without shrinkage, thereby producing less biased estimates.

Heinze et al. (2018) also mention the dependency of lasso on the scale of the variables in the feature space. As mentioned by Tibshirani (1996), the standard approach to applying lasso specifies that the covariates be standardised to have unit variance beforehand. Indeed, many software packages perform automatic standardisation internally and then return coefficients back to the user in their original scale. However, this might not always be optimal. Consider, for example, a problem that consists of both continuous and binary inputs, where continuous variables might encompass varying degrees of skewness and binary inputs could have different degrees of balance. The need to standardise the variables before subjecting them to the model also

imposes additional data manipulation requirements on the user. In the case of inexperienced users, the requirement of standardisation might not even be apparent at all. The dependency of lasso on the scale of the inputs can also be linked to the degree of bias inherent to coefficients produced by the model. Since the model delivers biased estimates, it might unnecessarily penalise effects where the true underlying parameter is large and incorrectly opt to include variables associated with smaller coefficients that may be less relevant to the target. For example, consider a count variable $X_1 \in G_1$, $G_1 = \{1, 2, 3, \dots\}$ and an amount variable $X_2 \in G_2$, $G_2 = \{\$10\,000, \$20\,000, \$100\,000, \dots\}$, both of which are in the modelling set. Notice that a one unit increase in X_1 will exert a substantially different effect on the prediction of $\mathbf{Y}$ as opposed to a one unit increase in X_2. It may therefore be reasonable to assume that β_{X_1} will be larger than β_{X_2}. Ultimately, this implies that one regression parameter might be penalised more than another simply because of the scale of the variable associated with it.

3.4.2 The elastic net

Zou and Hastie (2005) introduce the elastic net – a penalised regression technique that incorporates regularisation similar to lasso and ridge regression. In fact, as will be shown shortly, the elastic net is a combination of the two approaches presented in Section 3.4.1. The authors' work is motivated by a pursuit to overcome three main shortcomings inherent to lasso, namely:

- For problems where the number of predictors is larger than the number of observations in the dataset ($p > n$ problems), lasso can only select at most n variables for inclusion in the final model. As noted by the authors, selecting at most n out of p variables is not ideal within the realm of certain applications, such as gene selection problems, where microarray data typically consists of a small number of samples (less than 100 in some cases) and thousands of genes that represent the predictors. Typically, many genes will exhibit high levels of correlation between one another, forming a group-like structure. An ideal model would therefore be able to successfully complete two tasks: 1) eliminate unimportant genes that do not add value and 2) simultaneously include groups of genes into the model by performing a so-called "grouped selection", which could result in the final model having an extremely large number of inputs. Lasso would perform especially poorly when faced with the second task.

- Following first point above, if there exists a group of variables among which pairwise correlations are very high, lasso will opt to include only one variable from this group and exclude all others. In most scenarios, lasso will often not care which variable is selected from a group.

- For traditional $n > p$ problems, where the number of observations exceed the number of

variables in the design matrix, empirical studies have shown that the predictive performance of lasso might be compromised if severe correlations exist between the predictors.

Ultimately, the goal of the elastic net is to perform as well as lasso while simultaneously addressing the inadequacies listed above. As with lasso, the model does automatic variable selection and continuous shrinkage while still being able to deal with and select correlated groups of variables. The authors metaphorically liken the elastic net to a stretching fishing net that retains all the big fish.

Assuming that the response vector is centered by having $\sum_i Y_i = 0$ and that the columns of the design matrix are standardised such that $\sum_i X_{ij} = 0$ and $\sum_i X_{ij}^2 = 0$, the naive elastic net is defined for any nonnegative λ_1 and λ_2 as

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda_2\|\beta\|_2^2 + \lambda_1\|\beta\|_1, \tag{3.22}$$

where

$$\|\beta\|_2^2 = \sum_{l=1}^{p} \beta_l^2, \tag{3.23}$$

$$\|\beta\|_1 = \sum_{l=1}^{p} |\beta_l|. \tag{3.24}$$

By letting $\alpha = \lambda_1/(\lambda_1 + \lambda_2)$, (3.22)–(3.24) become the following optimisation problem:

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2, \tag{3.25}$$

subject to

$$(1 - \alpha)\|\beta\|_1 + \alpha\|\beta\|_2^2 \leq t. \tag{3.26}$$

Equation (3.26) is referred to as the elastic net penalty, which is a convex combination of the lasso and ridge penalties. When $\alpha = 1$, the naive elastic net becomes ridge regression. For all $\alpha \in [0, 1)$, the naive elastic net function is singular, meaning that it has the characteristics of the lasso and ridge regression.

In order for the elastic net to accommodate the grouping effect that may exist amongst variables in the design matrix (such as gene selection in microarray data), the authors consider the following penalised method:

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda J(\beta), \tag{3.27}$$

where $J(\cdot) > 0 \ \forall \ \beta \neq 0$.

The model considers the grouping effect of variables when the regression coefficients for a

subset of correlated predictors are roughly the same. In the extreme situation where one variable is exactly equal to another, the regression coefficients should be identical. Therefore, assuming that $X_i = X_j$ for some $i, j \in \{1, \ldots, p\}$ it follows that $\hat{\beta}_i = \hat{\beta}_j \; \forall \; \lambda > 0$ if $J(\cdot)$ is strictly convex. Furthermore, if $J(\beta) = \|\beta\|_1$ then $\hat{\beta}_i \hat{\beta}_j > 0$ and $\hat{\beta}^*$ is a minimiser of (3.27) where

$$\hat{\beta}_l^* = \begin{cases} \hat{\beta}_l & \text{if } l \neq i \text{ and } l \neq j, \\ (\hat{\beta}_i + \hat{\beta}_j)s & \text{if } l = i, \\ (\hat{\beta}_i + \hat{\beta}_j)(1-s) & \text{if } l = j, \end{cases}$$

for $s \in [0, 1]$.

Strict convexity will guarantee the grouping effect when two or more predictors are an exact replica of one another. While the lasso does not have a unique solution, the elastic net penalty is strictly convex when $\lambda_2 > 0$.

Additionally, if $\hat{\beta}_{\lambda_1 \lambda_2}$ is the elastic net solution, assuming that $\hat{\beta}_i \hat{\beta}_j > 0$, then the following distance metric is defined:

$$D_{\lambda_1 \lambda_2}(i, j) = \frac{1}{\mathbf{Y}_1} |\hat{\beta}_i - \hat{\beta}_j|, \tag{3.28}$$

where

$$D_{\lambda_1 \lambda_2}(i, j) \leq \frac{1}{\lambda_2} \sqrt{2(1-\rho)}. \tag{3.29}$$

The value $\rho = \mathbf{X}_i^T \mathbf{X}_j$ is the sample correlation measured between the two design matrix columns.

The metric $D_{\lambda_1 \lambda_2}(i, j)$ encapsulates the difference between the coefficient paths of X_i and X_j. If the two predictors are highly correlated, the difference metric is close to zero. The upper bound shown in (3.29) provides a quantitative representation of the grouping effect. Alternatively, lasso cannot accommodate a grouping effect.

While promising, the authors note that the naive elastic net only performs effectively if it is either very close to ridge regression or very close to lasso, which implies that the proposed model does not add significantly more value over existing procedures. Due to the construct of the penalty function, the model imposes a double amount of shrinkage on the coefficient estimates, which in turn introduces unnecessarily extra bias into model. In an attempt to combat this, the authors formulate the elastic net, which is based on a scaling transformation applied to the naive elastic net. The solution vector is given by

$$\beta^{elastic\ net} = (1 + \lambda_2)\beta^{naive\ elastic\ net}. \tag{3.30}$$

By utilising (3.30), the model is still able to perform variable selection whilst undoing unnecessary shrinkage. The use of $(1 + \lambda_2)$ as a scaling factor is motivated by considering the decomposition

of the ridge operator. Assuming that all predictors in the design matrix are standarised, the following matrix is obtained:

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & \rho_{12} & \dots & \rho_{1p} \\ \vdots & \ddots & & \vdots \\ & & 1 & \rho_{p-1,p} \\ \rho_{p1} & \dots & & 1 \end{bmatrix},$$

where ρ_{ij} is the sample correlation. As described earlier, the ridge estimate for a given λ_2 is given by $\hat{\beta}^{ridge} = \mathbf{RY}$. The matrix $\mathbf{R} = \left[\mathbf{X}^T\mathbf{X} + \lambda_2\mathbf{I}\right]^{-1}\mathbf{X}$ can be rewritten as

$$\mathbf{R} = \frac{1}{1+\lambda_2}\mathbf{R}^* = \begin{bmatrix} 1 & \frac{\rho_{12}}{1+\lambda_2} & \dots & \frac{\rho_{1p}}{1+\lambda_2} \\ \vdots & \ddots & & \vdots \\ & & 1 & \frac{\rho_{p-1,p}}{1+\lambda_2} \\ \frac{\rho_{p1}}{1+\lambda_2} & \dots & & 1 \end{bmatrix}\mathbf{X}^T,$$

where $\mathbf{R}^*$ is the OLS operator which is shrunk by $1/(1+\lambda_2)$. This causes ridge regression to have the desired grouping effect. While ridge regression requires the $1/(1+\lambda_2)$ shrinkage to control the variance of the model, the elastic net relies on the lasso penalty to control both shrinkage and variable selection. This implies that the model does not need the shrinkage step required by ridge regression. Consequently, the elastic net solution vector is given by

$$\beta^{elastic\,net} = \min_{\beta}\ \beta^T\left(\frac{\mathbf{X}^T\mathbf{X} + \lambda_2\mathbf{I}}{1+\lambda_2}\right)\beta - 2\mathbf{Y}^T\mathbf{X}\beta + \lambda_1\|\beta\|_1. \tag{3.31}$$

From (3.31) it follows that

$$\beta^{lasso} = \min_{\beta}\ \beta^T\left(\mathbf{X}^T\mathbf{X}\right)\beta - 2\mathbf{Y}^T\mathbf{X}\beta + \lambda_1\|\beta\|_1, \tag{3.32}$$

which means that the elastic net is a stabilised version of the lasso.

The elastic net model is implemented in a two-stage fashion, whereby the ridge regression coefficients are first obtained for a grid of λ_2 values, after which lasso shrinkage is performed. This means that for each λ_2, the elastic net is equivalent to a lasso model fit. The authors develop an algorithm called the LARS-EN in order to produce coefficient estimates in an efficient manner, which is based on the LARS algorithm by Efron et al. (2004). By proving that the lasso solution paths grow in a predictable piecewise linear fashion, Efron et al. (2004) show that the entire lasso solution path can be obtained using the same computational effort applied during an OLS fit. This subsequently allows the authors to implement the LARS algorithm to find the elastic net solution path with the same computational effort of an OLS model, given that the model is

simply a lasso problem for a given λ_2. Empirical studies performed on real and simulated data showed that optimal solution paths are achieved at an early stage of the LARS-EN algorithm, meaning that early stopping might aid the computational efforts required by the model. In particular, if the algorithm is stopped after m steps, it requires $O(m^3 + pm^2)$ operations.

As is the case with lasso and ridge regression, the elastic net requires parameter tuning. While the model is defined in terms of λ_1 and λ_2, the authors note that alternative choices regarding tuning parameters exist. Recall that the lasso problem can be defined in terms of (3.19)–(3.20). As mentioned in Section 3.4.1, an index parameter $s = t/\sum \beta_l$, where $s \in [0, 1]$, can be utilised for tuning the lasso model. Therefore, the elastic net can subsequently be parameterised by using (λ_2, s) or (λ_2, t). It is also noted that the use of s instead of t might be advantageous, given that s always lies between zero and one.

Like lasso and ridge regression, k-fold cross-validation serves as a popular method for hyper-parameter tuning during the elastic net fit (in particular, Zou and Hastie (2005) appear to favour tenfold cross-validation). In a typical scenario, the fitting of the elastic net starts off by considering a fairly small grid of values for λ_2. For each λ_2, the lasso solution path is calculated using the LARS algorithm. The corresponding lasso tuning parameter (λ_1, s or t) is selected via cross-validation, after which the λ_2 resulting in the smallest prediction error is chosen. The authors mention that such an approach is computationally efficient in the traditional $n > p$ case. Alternatively, a final model can be obtained in a relatively reasonable amount of time for $p > n$ problems (albeit not as fast as with $n > p$ datasets) as long as the number of predictors p are manageable. However, in order for model fitting to be less computationally expensive, early stopping recommended for $p > n$ settings.

The elastic net is associated with similar benefits inherent to lasso and ridge regression, given that the technique is an combination of the two penalties. Furthermore, the model also has the following added advantages:

- Like the lasso, the model facilitates variable selection by producing a sparse solution, thereby addressing one of the main drawbacks of ridge regression. However, as opposed to lasso, it is also able to handle $p > n$ problems, capable to deal with correlated predictor variables and can incorporate group structures that exist within the feature space. In summary, the model achieves what Zou and Hastie (2005) set out to do: produce a regularised regression model that overcomes the obstacles faced by lasso and ridge regression while simultaneously keeping the attractive properties associated with both approaches.

- Empirical evidence obtained from real-word and simulated data have shown that the elastic net outperforms various alternative techniques such as OLS, lasso and ridge regression, by a significant margin. While not disregarding the fact that the model is the most accurate estimator in most $n > p$ settings, the technique consistently produces the lowest prediction

error for problems where subsets of highly correlated variables exist or where $p > n$. One such an example that is used by the authors is the leukaemia dataset of Golub et al. (1999).

- In most cases, but especially for $n > p$ problems, the elastic net enjoys the computational benefits of lasso and is considered to be less resource intensive than best subset selection.

While the elastic net model shares the same advantages associated with its lasso and ridge regression counterparts, it also suffers from the same shortcomings. These include the introduction of bias into the model, the dependency of the coefficient estimates on the scale of the input variables, lack of interpretability of the effects and the assumption that the predictor variables are standardised before entering the model. However, the following additional concerns are also raised:

- As noted earlier in this section, the naive elastic net incurs double the amount of shrinkage than lasso or ridge regression and requires a scaling transformation to overcome this phenomenon. Specifically, the elastic net performs particularly poorly in the absence of scaling if its parameters are not set up in such a way that it closely resembles ridge regression.

- While the LARS-EN algorithm for finding the solution paths of the model is very efficient in most cases, Zou and Hastie (2005) acknowledge that the computational burden of the model increases considerably for $p > n$ problems and that early stopping of the algorithm is often required.

- The model incorporates twice as many shrinkage parameters than lasso or ridge regression, which means that parameter tuning via cross-validation needs to occur on a two-dimensional surface. This, in turn, requires more iterations of the model to be executed for disjointed sets of tuning parameters. For example, for each fixed λ_2 taken from a grid of λ_2 values, the process of performing k-fold cross-validation is computationally equivalent to k OLS regression model fits.

- As is the case with lasso and ridge regression, the elastic net requires the user to select the most optimal model from an array of models that were fitted using a range of different λ_1 and λ_2 values. In turn, this introduces a certain level of subjectiveness into the model selection process. Consider, for example, a scenario where a select few (λ_1, λ_2) combinations produce several models with similar prediction accuracies. This might result in two individual users selecting different models as their final choice. Since the elastic net has more tuning parameters than lasso and ridge regression, it might suffer from an exaggerated level of subjectiveness.

Ultimately, the elastic net appears to be a formidable variable selection technique that attempts to exploit the benefits associated with both lasso and ridge regression while overcoming some of

the disadvantages inherent to the two aforementioned approaches. However, the modeller must then face increased levels of complexity and subjectiveness when deciding on a final model.

3.4.3 The MCP and SCAD penalties

To address some of the shortcomings inherent to convex optimisation methods like lasso and the elastic net, various nonconvex penalties for regression have been proposed. In particular, two popular nonconvex penalised regression models that are often cited in literature include the *minimax concave* or MCP penalty (Zhang, 2010) and the *smoothly clipped absolute deviation* or SCAD penalty (Fan and Li, 2001). The fact that lasso suffers from biased estimates and that subset selection can become computationally inefficient serve as motivation for Zhang (2010) to formulate the MCP, noting that the minimax concave penalty does not suffer from any of the aforementioned drawbacks and can also be applied to $p > n$ problems (recall that lasso struggles with $p > n$ datasets). A similar motivation is given for the use of SCAD. Both SCAD and MCP regression models exhibit the oracle property, which implies that if the correct regularisation parameters are chosen, the resulting estimates will perform as well as if the correct underlying model was known in advance, i.e. the analyst or user had previous knowledge of which coefficients should be set exactly equal to zero (Fan and Li (2001); Breheny and Huang (2011)). Recall the penalised regression model in (3.27) which is given by

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + J_\lambda(\beta),$$

where $J(\cdot)$ is a penalty function that is greater than zero. For lasso, the penalty is $J_\lambda(\beta) = \lambda\|\beta\|_1 = \lambda\sum_{l=1}^{p}|\beta_l|$. In the case of MCP, the penalty function is defined on $[0, \infty)$ and is expressed as

$$J_{\lambda,\gamma}(\beta) = \begin{cases} \lambda|\beta| - \frac{\beta^2}{2\gamma} & \text{if } |\beta| \leq \gamma\lambda, \\ \frac{1}{2}\gamma\lambda^2 & \text{if } |\beta| > \gamma\lambda, \end{cases} \tag{3.33}$$

where $\lambda \geq 0$ and $\gamma > 1$.

In the case of SCAD, the penalty function is also defined on $[0, \infty)$ and can be written as

$$J_{\lambda,\gamma}(\beta) = \begin{cases} \lambda\beta & \text{if } |\beta| \leq \lambda, \\ \frac{\gamma\lambda|\beta| - \frac{1}{2}(\beta^2+\lambda^2)}{\gamma-1} & \text{if } \lambda < |\beta| \leq \gamma\lambda, \\ \frac{\lambda^2(\gamma^2-1)}{2(\gamma-1)} & \text{if } |\beta| > \gamma\lambda, \end{cases} \tag{3.34}$$

where $\lambda \geq 0$ and $\gamma > 2$.

Both penalties are very similar to one another and initially start off by applying the same rate of penalisation as lasso. However, at some point this rate of penalisation is reduced to zero

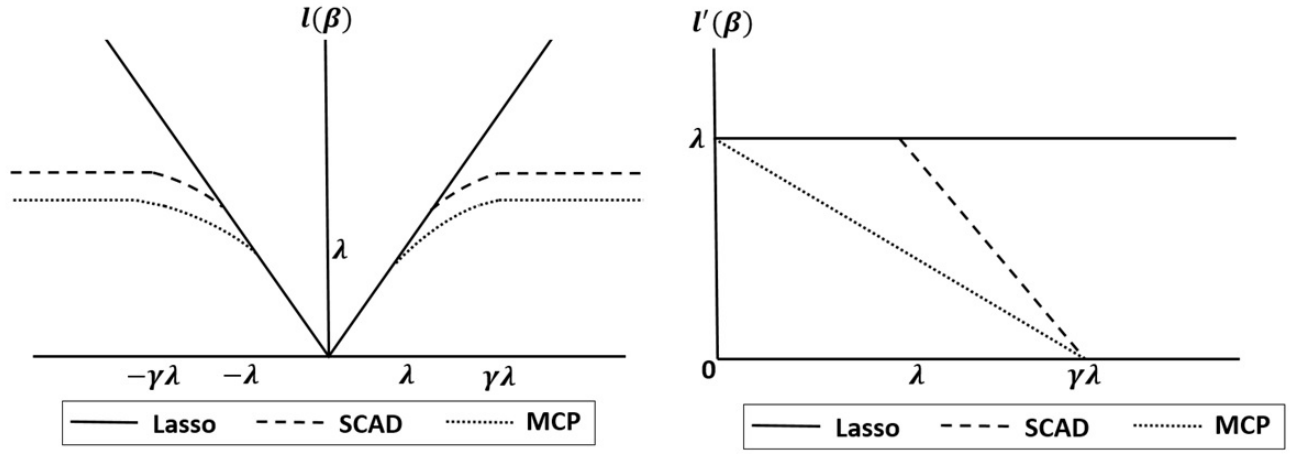


Figure 3.3: Lasso vs MCP and SCAD

as β becomes larger. For MCP, this point is where $|\beta| > \lambda\gamma$. For SCAD, the rate of penalisation remains flat for a while before it is reduced, while MCP relaxes the penalisation immediately. This results in larger coefficients not being penalised as severely as in lasso, which eliminates a significant portion of bias inherent to estimates produced by lasso models. The parameter λ controls the degree of regularisation (like in lasso and ridge regression) whereas γ controls how fast the rate of penalisation goes to zero. For MCP, if $\gamma \to \infty$, the model becomes the lasso, which is referred to as a soft-thresholding estimate. Alternatively, as γ approaches 1, the penalty performs like an L_0-penalty, which causes the model to behave in the same way as subset selection. In turn, subset selection is equivalent to hard-thresholding. The same applies to SCAD in the soft-thresholding case where $\gamma \to \infty$. However, SCAD does not exhibit the hard-thresholding property when $\gamma \to 2$. Therefore, both MCP and SCAD are said to be equivalent to "firm-thresholding" by bridging the gap between lasso and subset selection (Breheny and Huang, 2011). Figure 3.3 serves as a visual summary of the work performed by Breheny and Huang (2015) and shows graphically how penalisation is applied by lasso, MCP and SCAD, respectively. Both the penalty functions and their first derivatives (which represent each function's gradient) are shown.

Like lasso, ridge regression and the elastic net, the set of parameters for SCAD and MCP are commonly found by use of generalised cross-validation or k-fold cross-validation, as suggested by Breiman (1995), Tibshirani (1996) and Fu (1998). In particular, Fan and Li (2001) seem to favour fivefold cross-validation. As with previous settings, the model is fit for a two-dimensional grid of γ and λ values, where the optimal (γ, λ) is chosen at the point where the cross-validation error is minimised. To ease the computational burden, Fan and Li (2001) propose the use of $\gamma = 3.7$ in the case of the SCAD penalty. Indeed, empirical results shown by the authors indicate that $\gamma = 3.7$ often yields very satisfactory results and may even produce smaller prediction errors

than models utilising the SCAD penalty where cross-validation was implemented. However, this is often only applicable to linear regression models, where $\mathbf{Y} \in \mathbb{R}$, and may not necessarily hold true for logistic regression applications. Specifically, SCAD and MCP penalties appear to pose various challenges when applied to regression models where the output is dichotomous, which will be listed shortly.

Results from empirical studies have shown that penalised regression models utilising SCAD and MCP may be regarded as formidable contenders within the context of variable selection. Several efficient algorithms exist for fitting penalised regression models with SCAD and MCP penalties, such as the *local linear approximation* or LLA algorithm of Zou and Li (2008) and the coordinate decent method proposed by Breheny and Huang (2011). While the LLA algorithm yields an objective function that can be optimised using the very efficient LARS approach, the coordinate decent method is considered to be much more efficient. The algorithm optimises the objective function with respect to one parameter at a time and iteratively cycles through all parameters until convergence is obtained. Each pass over the set of parameters requires $O(np)$ operations. Since the computational expense of the coordinate decent method only increases linearly with p, it means that it can be applied even when p is very large. Another popular method for estimating regression parameters when using MCP and SCAD is *SparseNet* , which was proposed by Mazumder et al. (2011) and implemented in R. Zhang et al. (2013) suggests the use of a concave conjugate for fitting regression models with the MCP penalty, noting that their method only requires the selection of one parameter, instead of two (as was seen with all MCP and SCAD algorithms discussed so far), thereby making their algorithm more efficient.

Simulations conducted by Fan and Li (2001) show that SCAD often outperforms (sometimes only marginally) other selection techniques such as lasso and best subset selection (however, the authors note that SCAD and best subset selection seem to produce very similar results). MCP and SCAD appear to perform especially well for larger samples and for datasets where the level of noise has been reduced. Empirical results obtained from Breheny and Huang (2011) illustrate that MCP and SCAD allow coefficient estimates to take on much larger values much more easily than lasso, which often leads to the model outperforming lasso when the true underlying coefficients are large. The two suggested penalties seem to serve as useful alternatives to analysts who wish to avoid resource-intensive subset selection without having to settle for biased models. Various authors, such as Breheny and Huang (2011), Zhang (2010) and Zhang et al. (2013) note that MCP tends to outperform SCAD, with SCAD falling somewhere in between lasso and MCP. Specifically, MCP produces much sparser solutions with lower prediction errors than lasso and SCAD.

As with most penalised regression models, MCP and SCAD also suffer from a number of weaknesses. Because the penalty functions are not convex, the models are not guaranteed to converge to a global optimum. The coordinate decent algorithm of Breheny and Huang (2011)

deals with the non-convexity of the penalties by altering the lower bounds of γ by utilising the minimum eigenvalue of the matrix $\frac{1}{n}\mathbf{X}^T\mathbf{X}$, thereby enforcing the algorithm to converge to a global optimum. Nevertheless, the authors note that neither the coordinate decent algorithm nor the LLA of Zou and Li (2008) are guaranteed to converge in the case of logistic regression. Specifically, the reweighting that takes place in coordinate-wise updates results in γ becoming difficult to interpret for logistic regression problems. Breheny and Huang (2011) propose an adaptive rescaling of the penalties in an attempt to resolve these issues, but note that the adaptively rescaled SCAD solution is not equal to the regular SCAD solution. In convex penalised methods like the lasso, convexity also ensures that the estimated coefficient vector is continuous with respect to λ. In the absence of convexity, this does not hold true, which means that a small change in the data could exert a potentially large influence on the estimated parameters. This often results in a set of estimates with high variance. Furthermore, the lack of continuity with respect to λ makes it increasingly more difficult to choose an appropriate value for λ during model fitting.

Since the purpose of SCAD and MCP is to reduce the bias of the estimates, a reduction in bias will lead to an increase in variance. Increased variance coupled with the aforementioned unstable estimates will most likely yield a much more unstable model[5].

The use of cross-validation or information criteria such as the AIC and BIC to determine appropriate values for λ and γ has its own drawbacks. Breheny and Huang (2011) have observed that the AIC and BIC tend to select local optima (instead of global optima) in nonconvex regions of the objective function in certain experiments where MCP and SCAD were applied. While cross-validation does not suffer from local optima, it remains a resource-intensive procedure to perform over a two-dimensional grid. In turn, this means that convergence may take an extensive amount of time to achieve due to non-convexity. This may impede the modeller's ability to extensively investigate the choice of γ, which might result in some practitioners opting to use default values that may not be appropriate for the particular problem being modelled. Particularly, both models are extremely sensitive to the choice of γ in the $p > n$ case, where an incorrect parameter choice can result in significantly inflated prediction errors.

Lastly, the authors note that both MCP and SCAD tend to overfit badly on noisy data – especially for logistic regression problems. The problem of overfitting is more pronounced for MCP than for SCAD, specifically when γ is close to one.

The coordinate decent algorithm developed by Breheny and Huang (2011) performs noticably well and attempts to combat the problem of non-convexity by suggesting diagnostics that can easily be computed from the data. These diagnostics can potentially indicate which regions of the coefficient paths are locally convex, thereby retaining some of the benefits of convexity. The

[5]A model is considered to be unstable within an optimisation context when many local optima exist instead of one well-defined optimum.

authors also address the shortcomings associated with the AIC and BIC by suggesting a hybrid approach between the BIC, cross-validation and the aforementioned calculated diagnostics in order to choose appropriate values for γ and λ. However, the successful implementation of the authors' suggestions depends on the evaluation thereof by the user, which in itself might suffer from a certain level of subjectiveness (as is the case with lasso, ridge regression and the elastic net). MCP and SCAD approaches are also marred with greater levels of complexity when compared to other penalised regression methods.

Notice that each regression modelling technique introduced so far can be formulated as some mathematical programming problem, i.e. there is an objective function that has to be minimised or maximised subject to a set of constraints. In Chapter 4, exact mathematical modelling approaches are discussed in more detail.

Chapter 4

Exact mathematical modelling approaches

In this chapter, concepts of exact mathematical modelling approaches are introduced. Specifically, linear programming problems and mixed integer linear programming problems are discussed, along with popular methods used for solving such problems.

4.1 Linear programming

4.1.1 Introduction to LPs

According to Brandimarte (2006) a finite dimensional problem is a linear programming problem (LP) when both the constraints and objective function can be written as affine functions. In turn, an affine function is a function that can be expressed as a linear function (or functions) plus a constant term or bias. In one dimension, the graph of an affine function is a straight line. In n dimensions, an affine function is visually represented as a hyperplane and is notated as

$$f(x_1, \ldots, x_n) = A_1 x_1 + \cdots + A_n x_n + c,$$

where the coefficients $A_1, \ldots, A_n$ can be scalar values or matrices. The constant term c can be a single scalar or a column vector. Given the interpretation of an affine function, a linear programming problem can be written in the following general form:

$$\min \sum_{j=1}^{n} c_j x_j, \tag{4.1}$$

subject to

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad \forall i \in E, \tag{4.2}$$

$$\sum_{j=1}^{n} d_{ij}x_j \leq e_i \quad \forall i \in I. \tag{4.3}$$

The problem in (4.1)–(4.3) can be expressed in matrix form as

$$\min \ \mathbf{c}^T\mathbf{x},$$

subject to

$$\mathbf{Ax} = \mathbf{b},$$
$$\mathbf{Dx} \leq \mathbf{e}.$$

Linear programming problems display two very important characteristics within the context of mathematical programming, each of which is explained in more detail in Sections 4.1.2 and 4.1.3.

4.1.2 LP models are both convex and concave

To understand convexity, consider the definition of a convex set, which is given below.

Definition 4.1.1. A set $S \subseteq \mathbb{R}^n$ is a convex set if for any two points x and y in S

$$\lambda x + (1-\lambda)y \in S \ \ \forall \ \lambda \in [0,1].$$

Simply put, a set S is convex if the line segment joining any two points x and y, where $x, y \in S$, is also in S. Brandimarte (2006) discusses examples similar to those found in Figure 4.1, which serves as an excellent visual example to convey the idea of a convex set.

In Figure 4.1, the set S_1 is a convex set, as per Definition 4.1.1, whereas the set S_2 is a non-convex set. The set S_3 is a discrete set, which is not convex, but it does possess a convex hull, which will be explained shortly.

Definition 4.1.2. A function $f : \mathbb{R} \to \mathbb{R}^n$ defined over the convex set $S \subseteq \mathbb{R}^n$ is a convex function on S if the following holds true for any two points x and y in S and for any $\lambda \in [0,1]$:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y). \tag{4.4}$$

If (4.4) is true with strict inequality for all $x \neq y$, the function is said to be *strictly convex*.

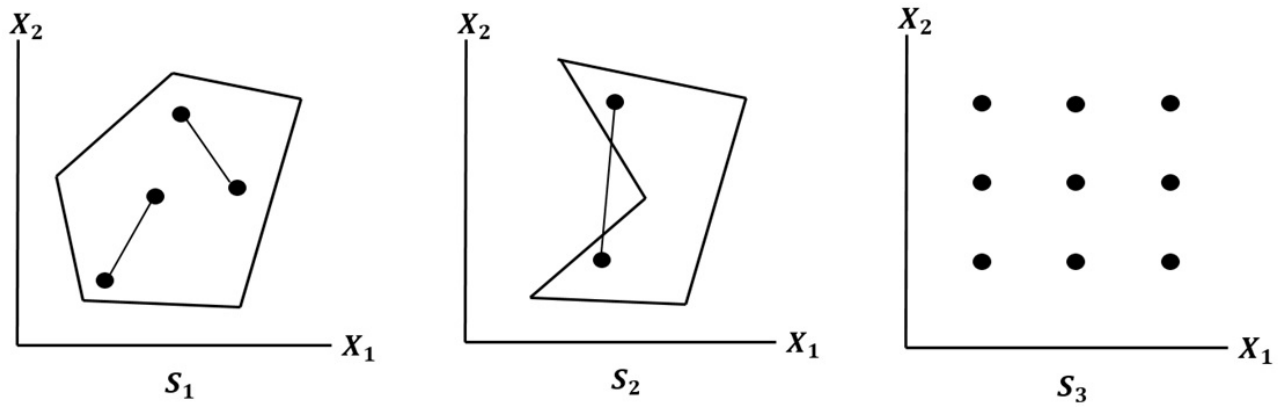


Figure 4.1: Convex and non-convex sets

Alternatively, a function f is concave if $-f$ is convex, or if

$$f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda)f(y). \tag{4.5}$$

Again, if (4.5) holds true with strict inequality, then the function is *strictly concave*. Figures 4.2 and 4.3 on the next page are visual examples of a convex and a concave function in one dimension.

A very important and attractive property of convex and concave functions is the fact that any local optimum $\hat{x}$ will also be a global optimum, x^*.

Definition 4.1.3. Consider an optimisation problem $\min f(x)$ where $x \in S \subseteq \mathbb{R}^n$. The solution x^* to the minimisation problem is said to be a global optimum if $f(x^*) \leq f(x)\ \forall\ x \in S$. Alternatively, for a maximisation problem, x^* is considered to be a global optimum if $f(x^*) \geq f(x)\ \forall\ x \in S$.

Simply put, if the problem is a minimisation problem over the region S, then the solution to the problem will be a global optimum if no other point in S produces a smaller objective value. The converse is true for a maximisation problem.

Intuitively, it follows that a solution is a local optimum if Definition 4.1.3 only holds true for a neighbourhood of x^* in S, i.e. in the case of a minimisation problem, $f(x^*) \leq f(x)\ \forall\ x \in D$, where $D \subseteq S$, but $f(x^*) \not\leq f(x)\ \forall\ x \in S$. The notation can easily be extended to a maximisation problem by swapping the inequality. Figure 4.4 shows two functions, one with both a local and global minimum and another with only a global minimum. Notice that the function with both a local and global minimum is not convex, whilst the other one is.

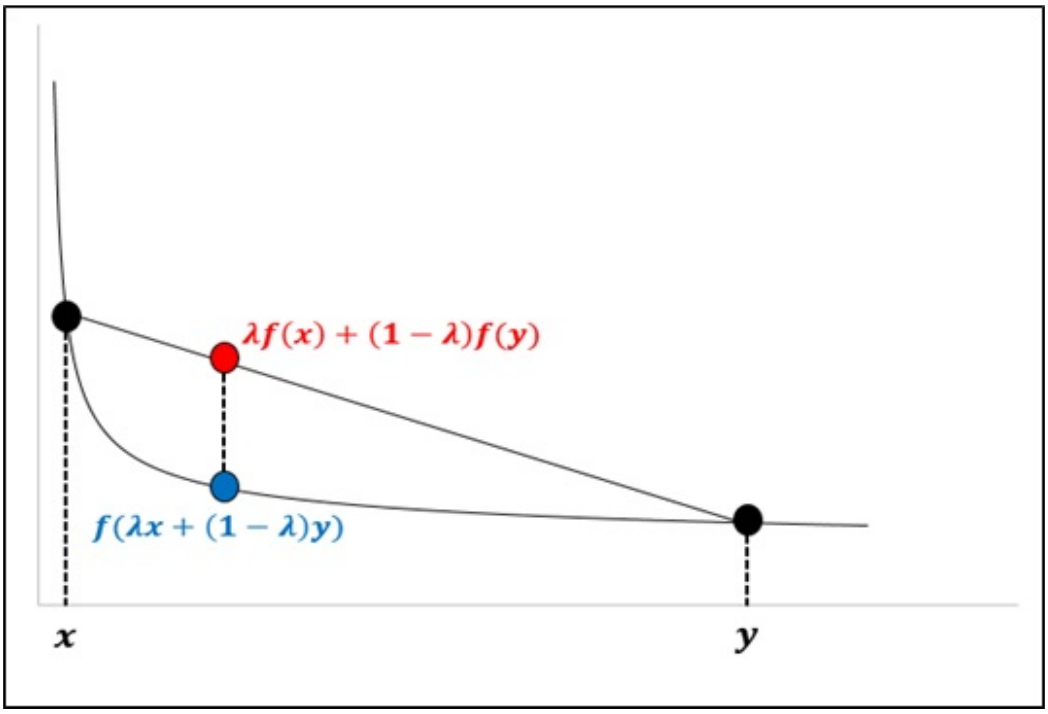


Figure 4.2: Visual example of a convex function

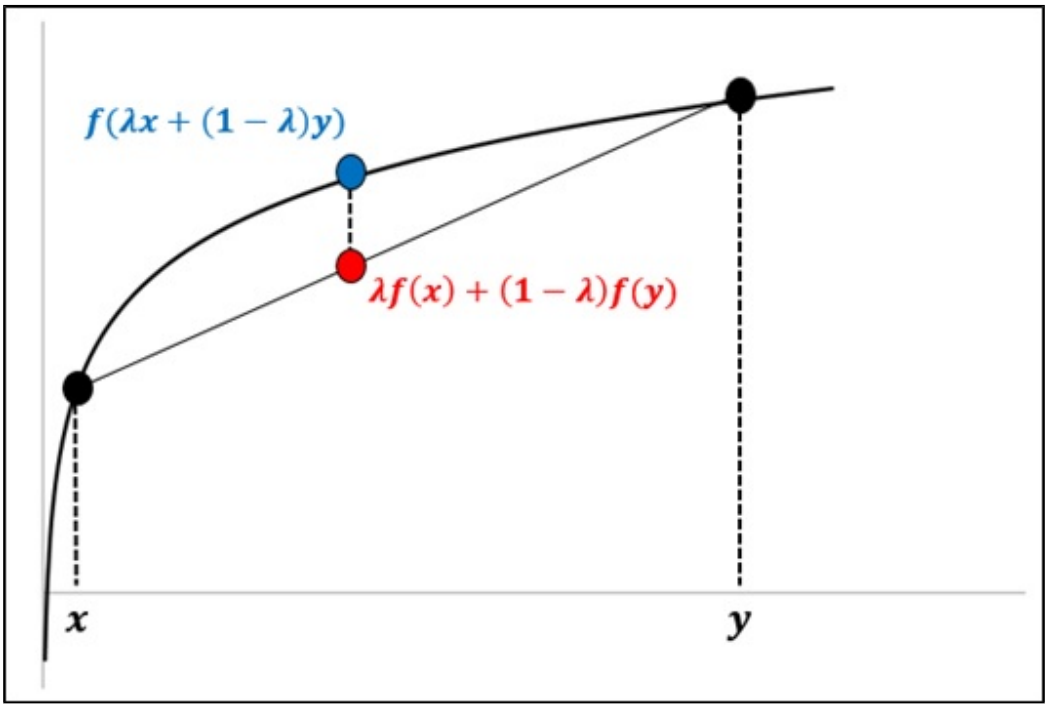


Figure 4.3: Visual example of a concave function

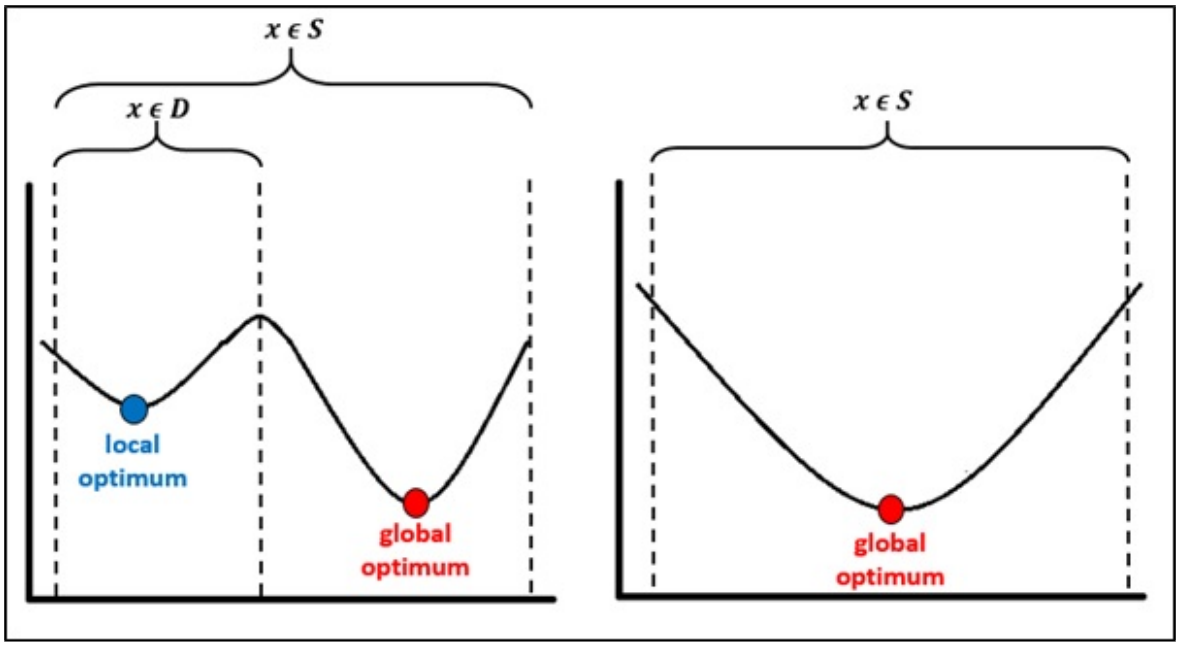


Figure 4.4: Local vs global optimum

Given the understanding of local and global optima, consider the following theorem:

Theorem 4.1. Let $f : \mathbb{R} \to \mathbb{R}^n$ be a convex function over the set $S \subseteq \mathbb{R}^n$. If x^* is a local minimum of f over S, then x^* is also a global minimum of f over S.

The proof for Theorem 4.1 has been well-documented in literature – see e.g. Beck (2014).

Proof. Let x^* be a local minimum and $y \in S$ be an arbitrary point in S, where $x^* \neq y$. Suppose that a small enough $\lambda > 0$ is chosen, noting that as $\lambda \to 0$ we find $f(x^*)$, such that

$$f(x^*) \leq f(\lambda y + (1 - \lambda)x^*).$$

Since f is convex, it implies that

$$f(\lambda y + (1 - \lambda)x^*) \leq \lambda f(y) + (1 - \lambda)f(x^*).$$

If follows that

$$f(x^*) \leq \lambda f(y) + (1 - \lambda)f(x^*).$$

Rearranging shows that

$$\lambda f(x^*) \leq \lambda f(y),$$

which gives

$$f(x^*) \leq f(y).$$

Since y is an arbitrary point in S, it shows that x^* is a global minimum of f. The same can be shown for a concave functon by writing out the above notation in terms of $-f$. □

Lastly, it needs to be shown that a linear programming problem is both convex and concave. In fact, the proof thereof is quite simple and straightforward. Recall that a linear programming problem can be expressed in terms of one or more affine functions. Consider, then, the following theorem and the subsequent proof (Ahmadi, 2016):

Theorem 4.2. Let $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$ be an affine function for any $\mathbf{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$. It follows that $f(\mathbf{x})$ is both convex and concave, but neither strictly convex nor strictly concave.

Proof. Since an affince function is a straight line in one dimension and a hyperplane in $n > 1$ dimensions, it follows that for all $\lambda \in [0, 1]$

$$\begin{aligned} f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) &= \mathbf{a}^T(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) + b \\ &= \lambda\mathbf{a}^T\mathbf{x} + (1-\lambda)\mathbf{a}^T\mathbf{y} + \lambda b + (1-\lambda)b \\ &= \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}). \end{aligned}$$

□

Specifically, affine functions are the only functions that are both convex and concave at the same time. Since a linear programming problem consists of a system of affine functions, it implies that such an optimisation problem will be convex and concave. In turn, this means that any solution to an LP will always be a global solution.

4.1.3 The feasible set forms a polyhedron where the optimal solution lies on one of the vertices

An important aspect of LPs to consider is the fact that the feasible set is a polyhedron, where the solution to the optimisation problem can be found at one of the extreme points. Consider, therefore, the definition of a polyhedron below:

Definition 4.1.4. Let $\mathbf{a}_i^T\mathbf{x} = b_i$ be a hyperplane in $\mathbb{R}^n$, where $b_i \in \mathbb{R}$ and $\mathbf{a}_i^T\mathbf{x} \in \mathbb{R}^n$ is a column vector. The hyperplane divides $\mathbb{R}^n$ into half-spaces via the inequalities $\mathbf{a}_i^T\mathbf{x} \leq b_i$ and $\mathbf{a}_i^T\mathbf{x} \geq b_i$. A polyhedron is then the set of points $P \subseteq \mathbb{R}^n$ that satisfies the set of linear inequalities

$$P = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} \geq \mathbf{b}\}.$$

This means that a polyhedron is the intersection of a finite collection of half-spaces. It should be noted that a polyhedron is a convex set, since it is the intersection of convex sets.

Definition 4.1.5. A polyhedron is bounded if there exists a positive number M such that

$$P = \{\mathbf{x} \in \mathbb{R}^n | -M \leq x_j \leq M,\ j = 1, \ldots, n\}.$$

A bounded polyhedron is known as a polytope. Following the definition of a polyhedron and bounded polyhedron, the definition of a convex hull is introduced before the extreme points of a polyhedron can be addressed.

Definition 4.1.6. The convex combination of n points $\mathbf{x} = \{x_1, \ldots, x_n\}$, $\mathbf{x} \in \mathbb{R}^n$, is defined as

$$\mathbf{x} = \sum_{j=1}^{n} \lambda_j x_j,$$

where

$$\lambda_1, \ldots, \lambda_n \geq 0,$$
$$\sum_{j=1}^{n} \lambda_j = 1.$$

Any set of points that are a convex combination of the points in the set $S \subseteq \mathbb{R}^n$ is known as the *convex hull* of S, denoted by $[S]$. If S in itself is a convex set, then $S \equiv [S]$. Specifically, the convex hull of any set S is the smallest convex set containing S or the intersection of all convex sets that encompass S. Recall the set S_3 in Figure 4.1. The set S_3 is discrete, which means that it is not convex. However, it does indeed have a convex hull, which is shown in Figure 4.5 (note that the convex hull is the grey area in the figure which is bounded by the black lines and coordinates). Discrete sets will be discussed in more detail in the following sections addressing mixed integer programming.

Given the explanations of a polyhedron, bounded polyhedron and a convex hull, the definition of an extreme point needs to addressed.

Definition 4.1.7. A point $\mathbf{x}$ is an extreme point of a polyhedron P if $\mathbf{x} \in P$ and it is not possible to write $\mathbf{x}$ as $\mathbf{x} = \lambda x_1 + (1 - \lambda)x_2$ for some $0 \leq \lambda \leq 1$, with $x_1, x_2 \in P$ and $x_1 \neq x_2$, i.e. $\mathbf{x}$ cannot be written as a convex combination of any other points in the set.

Alternatively, any point $\mathbf{x}$ that is *not* an extreme point of P can be expressed as a convex combination of the extreme points in P, or $\mathbf{x} = \sum_{j=1}^{\tau} \lambda_j x_j$, where τ is the number of extreme points, $\lambda_j \geq 0$ and $\sum_{j=1}^{\tau} \lambda_j = 1$. In any polytope, there only exists a finite number of extreme points $x_1, \ldots, x_\tau$. Specifically, in the case of a bounded polyhedron, the polytope itself is the convex hull of the extreme points.

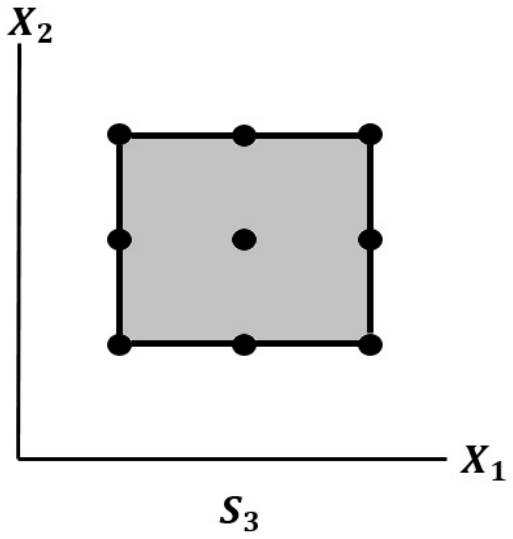


Figure 4.5: Convex hull of a set

Figure 4.6 summarises most of the concepts introduced in this section. The figure displays, in two dimensions, the convex hull of the feasible set (the area that is shaded in grey and bounded by the dashed lines) which is also a bounded polyhedron or polytope. The dashed lines represent the affine functions dividing the space $\mathbb{R}^2$ into half-spaces. Lastly, the coordinates at the corners of the feasible region represent the extreme points of the polyhedron.

Recall that the assertion was made that the optimal solution to an LP problem lies at one of its vertices, which is also one of the extreme points.

Definition 4.1.8. A constraint of the form $\mathbf{a}_i^T\mathbf{x} \leq b_i$, $\mathbf{a}_i^T\mathbf{x} = b_i$ or $\mathbf{a}_i^T\mathbf{x} \geq b_i$ is said to be binding at $\mathbf{x}^*$ if $\mathbf{a}_i^T\mathbf{x}^* = b_i$. Naturally, equality constraints will always be binding.

Definition 4.1.9. A point $\mathbf{x} \in \mathbb{R}^n$ is a vertex of a polyhedron P if it is feasible, i.e. $\mathbf{x} \in P$, and there exists n linearly independent constraints that are binding at $\mathbf{x}$.

Definition 4.1.10. In an n-dimensional programming problem, a point $\mathbf{x} \in \mathbb{R}^n$ is called a basic solution if n constraints are binding at that point. Furthermore, a point $\mathbf{x}$ is called a basic feasible solution if it is a basic solution which is also feasible.

Proofs involving Definitions 4.1.8, 4.1.9 and 4.1.10 which show that a basic feasible solution is equal to a vertex and that a vertex is equal to an extreme point (which also implies that a basic feasible solution is equal to an extreme point) are provided by Karger (2004).

Ultimately, the optimal solution to a linear programming problem lies on the vertex of the polyhedron, which is also an extreme point and a basic feasible solution. The fact that the optimal solution is located at one of the extreme points is an important attribute of LPs which

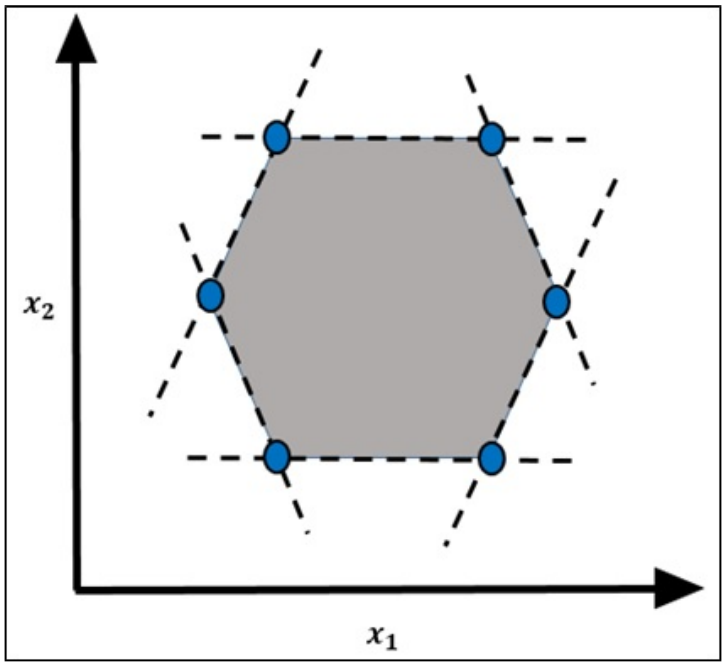


Figure 4.6: Example of a bounded polyhedron

is exploited by many solvers, as it guarantees convergence must faster, since only the vertices have to be evaluated.

4.1.4 Solving linear programming problems: the simplex method

The simplex method is arguably one of the most notable and popular methods used for solving linear programming problems today (Bixby, 2012). Fourer (2014) provides a brief yet adequate summary of the simplex method and its origins, which is discussed next. The method's roots can be traced back to period between 1947 and 1949, when it was first conceived by George Dantzig. Dantzig (1951) explains that a problem can be transformed to one wherein a linear form of nonnegative variables can be maximised subject to a set of linear equalities and provides a proof for its convergence. Cooper et al. (1953) then proceeded to illustrate the use of the popular simplex tableau – an easy and straightforward method for finding a solution via the simplex method, which can be solved manually (without the assistance of a computer) when dealing with small scale linear programming problems. The simplex tableau arranges all of the variables and equations in a convenient table which is then solved through a series of Gaussian reduction schemes, whereby independent row equations can be set equal to multiples of themselves and subtracted from one another to express the problem in canonical form (Luenberger and Ye, 2016). The tableau systematically repeats this set of equations until a solution is found. An overabundance of examples illustrating the use of the simplex tableau exist today and are readily available through video tutorials or in a variety of text books. Dantzig (1953b) and Dantzig

et al. (1954) proceed to introduce additional computational advances for the simplex method, wherein, amongst other benefits, the main advantages included smaller simplex tableau updates and an algorithm that makes use of sparse operations to take advantage of the large number of zeros in the tableau. Dantzig (1953a) break further ground by fully making use of the simplex matrix's sparse representation and introduce an algorithm that is practically computable. The simplex method gained popularity from the 1960's onwards, with works by authors such as Dantzig (1963) and Orchard-Hays (1968) being incorporated into text books and courses taught at tertiary institutions.

The simplex method is based on a simple yet intuitive mathematical idea, which can be summarised as follows:

Proposition 4.1.1. Let v be a vertex on the bounded polytope P where $f(\mathbf{x}^v)$ is the objective function value for the LP, with $\mathbf{x}^v$ the coordinate vector at v. Furthermore, suppose that the LP is a maximisation problem (for a minimisation problem, the inequalities can simply be reversed). While there exists a neighbouring vertex of v, called v^*, where $f(\mathbf{x}^v) < f(\mathbf{x}^{v^*})$, then set $v = v^*$, with $\mathbf{x}^{v^*}$ being the new solution and $f(\mathbf{x}^{v^*})$ being the new objective value for the LP.

Recall that it was shown that the optimal solution for an LP lies at one of the extreme points of its feasible region, which is also a vertex. Ultimately, the simplex method exploits this trait of LP problems by evaluating the objective at the current vertex and then moving to a neighbouring vertex if the objective function value can be improved. By doing so, the simplex method systematically progresses in the direction of the optimal value of the LP, until no neighbouring extreme points can improve the function value.

Dasgupta et al. (2008) outlines the simplex method in a thorough yet easy-to-follow manner, which will be explained next. Firstly, it is important to note that if there exists an LP consisting of n variables, then at least n linear constraints are needed to obtain a unique global optimum. Again, consider the standard form of an LP, which is given by

$$\max \ \mathbf{c}^T\mathbf{x},$$

subject to

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}.$$

The set of constraints $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ can be written as the following set of linear inequalities:

$$\begin{aligned} \mathbf{a}_1\mathbf{x} &\leq b_1, \\ &\vdots \\ \mathbf{a}_m\mathbf{x} &\leq b_m, \end{aligned}$$

where m is the number of rows in the constraint matrix.

Given that a vertex in $\mathbb{R}^n$ consists of n linear inequalities that are binding, the definition of a neighbouring vertex is as follows:

Definition 4.1.11. Two vertices, v_1 and v_2, in $\mathbb{R}^n$ are considered neighbours if v_1 and v_2 have $n-1$ linear inequalities in common.

Now that a neighbouring vertex has been defined, the inner workings of the simplex method can be explained next. Based on the summary of the simplex method given in Proposition 4.1.1, consider then the tasks that need to be performed by the algorithm, which is:

1. If the current vertex v is optimal, then stop.

2. If the first point is not true, then determine which vertex to progress to next.

The above two tasks are much easier to perform if the vertex being evaluated at the current step of the simplex method is at the origin, i.e. if $\mathbf{x}^v = (x_1 = 0, x_2 = 0)$ in $\mathbb{R}^2$. To explain this, consider the LP in Proposition 4.1.1, where the objective is to maximise $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{x} \geq \mathbf{0}$. Next, let the origin be feasible. If the origin is feasible, it will also be a vertex, since all inequalities are binding. It is then noted that the origin is only globally optimal if $\mathbf{c} \leq \mathbf{0}$. Due to having $\mathbf{c} \leq \mathbf{0}$ and $\mathbf{x} \geq \mathbf{0}$, this result should be straightforward, since there is no possible way to obtain a better objective function value if $\mathbf{x} > \mathbf{0}$. However, if some entries of $\mathbf{c}$ are indeed greater than zero, then it is known that the origin is not optimal, since $\mathbf{c}^T\mathbf{x}$ can be improved by increasing one or more of the entries in $\mathbf{x}$. Therefore, to perform the second task listed above, the simplex method moves along the direction of one of the hyperplanes in $\mathbb{R}^n$ by continuously increasing some of the x_i's for which $c_i > 0$. This is done by "releasing" the inequalities[1] given by the constraints $x_i \geq 0$ and inflating the x_i's until another constraint is met with strict equality, i.e. becomes binding. The simplex method will again be at a point where n inequalities are binding, which means that a new vertex has been discovered.

When the simplex method has reached the vertex v^{+1}, where v^{+1} is not the origin, it will then transform v^{+1} such that it becomes the origin by shifting the coordinates $\mathbf{x}$ (this transformation encapsulates the same logic used by the Gaussian operations that are performed in the simplex tableau). The newly obtained coordinates will be at a distance $\mathbf{y}$ from the n hyperplanes that are defined by the n inequalities that intersect at v^{+1}. Specifically, for the inequality $\mathbf{a}_i\mathbf{x} \leq b_i$, the distance y_i is given by $y_i = b_i - \mathbf{a}_i\mathbf{x}$. Since the distances in $\mathbf{y}$ are a linear function of $\mathbf{x}$, it implies that the inverse relationship can be used to express $\mathbf{x}$ as a function of $\mathbf{y}$. This means that the LP can be rewritten in terms of $\mathbf{y}$. It should be noted that the objective function value of the LP remains the same, with only the coordinates that change. The rewritten LP with $\mathbf{y}$ as its coordinates then has the following characteristics:

[1]Releasing an inequality means that the constraint is not binding anymore, i.e. the values of the coordinates in the constraint are changed such that it is no longer met with strict equality.

- It has the inequalities $\mathbf{y} \geq \mathbf{0}$, which is the transformed version of the inequalities containing $\mathbf{x}$ that intersect at v^{+1}.
- The vertex v^{+1} now becomes the origin within the $\mathbf{y}$ coordinate system.
- The objective function becomes $\max \mathbf{c}_{v^{+1}} + \mathbf{c}^*\mathbf{y}$, with $\mathbf{c}_{v^{+1}}$ the value of the objective function at v^{+1} and $\mathbf{c}^*$ the transformed coefficient vector.

This means that the simplex method is at the origin once more and can start again with the first task. As discussed earlier, the method will stop once all of the coefficients at the newly defined origin are negative or equal to zero. Alternatively, if the current vertex is not optimal, the simplex method repeats this process and moves along the direction of one of hyperplanes until another inequality is found to be binding and a new vertex is discovered. A detailed example of the simplex method wherein the aforementioned transformations are systematically applied in a step-by-step fashion until an optimal solution is found, is provided by Dasgupta et al. (2008).

A specific problem faced by the simplex method is the presence of degenerate vertices. A degenerate vertex is one where the extreme point is defined by the intersection of more than n hyperplanes that form a polyhedron in $\mathbb{R}^n$. As explained by Luenberger and Ye (2016), linear programming problems – including those handled by the simplex method – are conveniently simplified by the assumption of nondegeneracy, which states that every feasible solution to the problem is a nondegenerate solution. Degenerate vertices pose an obstacle in the sense that some of its neighbours have identical values, which causes the simplex method to consider the degenerate point as an optimal solution (even if it is not). Notice that the straightforward simplex method cannot conclude in such a case, as it will jump from one vertex to the next without recording any improvement in the objective value, thereby initiating an infinite loop. The problem is averted by a method know as *perturbation*, which involves modifying the simplex method slightly such that the value b_i in $\mathbf{a}_i\mathbf{x} \leq b_i$ now becomes $b_i^{new} = b_i \pm \epsilon_i$ for some very small value ϵ_i. The LP should not change drastically, since ϵ_i is almost negligible. However, it allows degenerate vertices to be split into more than one extreme point, where the new vertices are no longer degenerate. Even though the newly created vertices will be very close to one another, they will allow the simplex method to travel from one vertex to the next without encountering the aforementioned challenges.

Another consideration that has to be made when executing the simplex method is determining the starting vertex of the algorithm. For small-scale problems in a low-dimensional space, this task is quite simple. However, for a larger LP in n dimensions, it becomes all but trivial. But, as Dasgupta et al. (2008) and Luenberger and Ye (2016) point out, finding the starting vertex can be seen as an LP as well. To explain this, consider once more an LP that is being maximised. In order to start, the LP needs to be rewritten in canonical form – that is, all the inequality constraints of the form $\mathbf{Ax} \leq \mathbf{b}$ need to become $\mathbf{Ax} = \mathbf{b}$. This is achieved by the use of

slack variables. For example, the inequality constraint $\mathbf{a}_i\mathbf{x} \leq b_i$ is transformed to an equality constraint by introducing the slack variable s_i and writing $\mathbf{a}_i\mathbf{x} + s_i = b_i$. Additionally, all b_i need to be nonnegative. Given these prerequisites, the new LP is then formed by:

1. Creating m variables, $s_1, \ldots, s_m$, where m is the number of inequalities in the LP.

2. Adding the variable s_i to the left-hand side of the i-th inequality such that it becomes $\mathbf{a}_i\mathbf{x} + s_i = b_i$.

3. Setting the objective function equal to $\min \sum_{i=1}^{m} s_i$.

Using the simplex method to solve the LP shown above, the starting vertex will be the one with $s_i = b_i \; \forall \; i$ where all other variables are set equal to zero. Specifically, if the optimum value of $\sum_{i=1}^{m} s_i$ is zero, it implies that all s_i's are zero and that the starting vertex of the original LP is found by simply starting at the optimum solution of the new LP and ignoring the s-variables. However, if the objective value of $\sum_{i=1}^{m} s_i$ is not zero, it means that the original LP is infeasible and the simplex method will stop.

Note that the simplex method cannot deal with unbounded problems and will cease as soon as the algorithm realises it. The concept of a unbounded problem is an important one. Consider, for example, an unbounded maximisation problem such as the one below:

$$\max \; f(\mathbf{x}) = 2x_1^2 + x_2 + c,$$

subject to

$$x_1 \geq 0, x_2 \geq 0.$$

Clearly, a larger objective function value for $f(\mathbf{x})$ will always be attainable by arbitrarily increasing either x_1 or x_2 or both, seemingly with no end in sight. Specifically, if $\mathbf{x}^*$ is the coordinate vector containing x_1 and x_2 at which the solution is found, then $\lim_{\mathbf{x}^* \to \infty} f(\mathbf{x}^*) = \infty$. The concept can easily be extended to a minimisation problem by reversing the inequalities.

Lastly, consider the execution time of the simplex method. The simplex method is somewhat of a paradox due to the fact that there exists other methods for solving LPs that are proven to be theoretically superior to the simplex algorithm, such as the Khachian's ellipsoid algorithm (Adejo and Okutachi, 2012) and Karmarkar's interior-point method (Nesterov and Nemirovski, 1994), both of which are hypothesised to deliver solutions in polynomial time. The simplex method, on the other hand, is not a polynomial-time algorithm. However, as noted by Dasgupta et al. (2008), the simplex method appears to outperform most other algorithms in practical applications. Clearly, this creates an environment where theory and practical examples contradict one another. To explain the execution time of the simplex method, consider the fact that a vertex is defined by n inequality constraints. Since its neighbouring vertices share $n-1$ of these inequalities, it implies

that each vertex can have at most nm neighbours, where m is the number of inequalities. To establish whether a vertex is indeed a true extreme point requires the method to solve a system of n equations for n variables and then subsequently checking if the result is a basic feasible solution. In turn, this consumes $O(n^3)$ time, with a time of $O(mn^4)$ per simplex iteration. However, recall the algorithmic approach illustrated earlier in this section wherein the coordinate system is continuously transformed to reflect a vertex at the origin. In fact, by utilising this transformation it can be shown that the result of the algorithm changes only slightly from one iteration to next, which leads to an execution time of $O(n[m+n])$ when expressing the new LP in terms of the distances $\mathbf{y}$ during each step. Since the new objective function is given by $\max \mathbf{c}_{v+1} + \mathbf{c}^*\mathbf{y}$, a new neighbouring vertex is found quite easily by selecting any $c_i^* > 0$ as an indication of the new direction to move in. It then becomes a lot easier to determine by how much each y_i can increase before one or more constraints are violated. Ultimately, this leads to an execution time of $O(mn)$ per simplex iteration. The value $UB = \binom{m+n}{n}$ serves as an upper bound on the number of iterations, since this is the maximum number of allowable vertices that can exist. However, notice that UB is exponential in n, which makes the simplex method an exponential-time algorithm. Indeed, there exists a special class of problems which require an exponential number of iterations when solved by the simplex approach – see e.g. Section 5.2 of Luenberger and Ye (2016). However, such problems have – at least for now – never been encountered in real-world applications, which partially explains why the simplex method enjoys such wide-spread popularity.

An example of solving a maximisation problem in $\mathbb{R}^2$ using the simplex method is available in Ahmadi (2016).

4.2 Mixed integer programming and mixed integer linear programming

4.2.1 Introduction to and inner workings of MIPs and MILPs

So far, the only problems that have been dealt with had $\mathbf{x} \in \mathbb{R}^n$, where the decision variables $x_i \in \mathbb{R}$ were allowed to be real-valued numbers that can take on any value (only positive values if $x_i \geq 0$) which may or may not include fractional values. However, as Smith and Taskin (2007) explains, the modeller is often faced with scenarios where it is practically impossible to accept fractional quantities as solutions. For example, a chief of staff at a hospital cannot hire 6.5 new nurses nor can a manufacturing plant order 100.4 crates to package its products. Therefore, the need for integer programming arises from problems where the quantities associated with the decision variables must be integer amounts. This requirement is often referred to as the integrality of quantities when addressing integer programming. Additionally, it is generally not

desired to simply use the rounded solutions of an LP as a possible solution to a MIP, as this might deliver sub-optimal solutions or cause certain constraints to be violated for large-scale optimisation problems. In general, an integer programming problem can be expressed in the following way:

$$\max \ \mathbf{c}^T\mathbf{x}, \tag{4.6}$$

subject to

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{4.7}$$

$$\mathbf{x} \in \mathbb{Z}^n \ \text{ or } \ x_i \in \mathbb{Z}. \tag{4.8}$$

In problem (4.6)–(4.8), notice that the solution to all decision variables must be whole numbers. If the decision variables can only be positive, then $x_i \in \mathbb{Z}^+$ which means that $x_i \in \{0, 1, 2, \dots\}$. A special case of integer programming problems exist where the decision variables are specified to be binary, meaning that the solution to each variable can only be a one or a zero, i.e. $x_i \in \{0, 1\}$. This can also be referred to as binary integer programming or "0-1" integer programming. A good example of a binary integer programming problem is the very popular knapsack problem. As explained earlier in this section when linear programs were discussed, recall that problems where decision variables can only take on values in a discrete set are neither convex nor concave. To grasp this concept, consider again the discrete set S_3 shown in Figures 4.1 and 4.5. The points in S_3 all represent possible values that the decision variables may take on. Since any value that is not exactly equal to one these coordinates is infeasible, it would imply that an arbitrary line spanning between any two coordinates will not be in the feasible set. However, as shown in Figure 4.5, it is still possible that the feasible set has a convex hull that can form a bounded polyhedron.

Integer programming problems can be extended to problems where only some of the variables are required to take on integer values, whilst others are free to assume any value in $\mathbb{R}$, which may include fractional solutions. These are commonly known as *mixed integer programming problems* or MIPs. Furthermore, when the problem is expressed in terms of a system of affine functions, such as those seen in LPs, it can then be referred to as a *mixed integer linear programming problem* or MILP. The terms mixed integer programming and mixed integer linear programming are often used as interchangeable jargon, with many referring to one as other. However, this is not entirely correct. While a MILP can always be seen as a MIP, it should be noted that nonlinear optimisation problems such as mixed integer quadratic problems or MIQPs also fall under the broader MIP-umbrella.

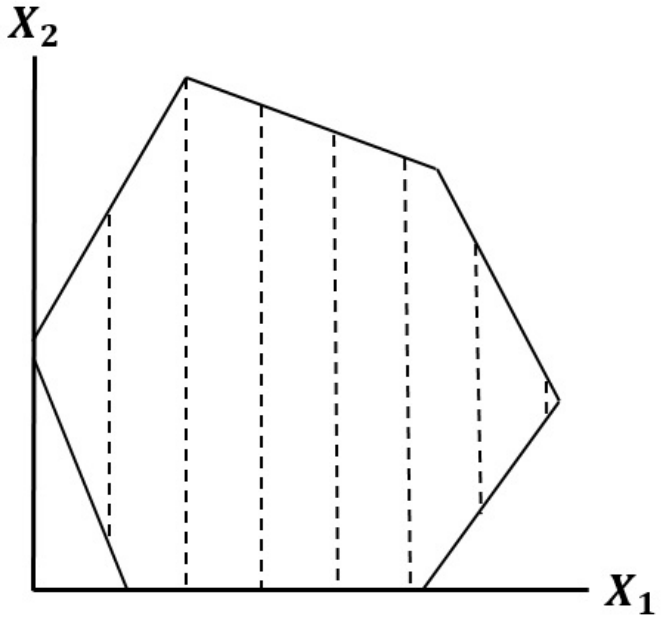


Figure 4.7: Graphical representation of a MILP in two dimensions

The general form of a mixed integer linear model can be written as follows (Junger et al., 2009):

$$\max \ \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y},$$

subject to

$$\begin{aligned} &\mathbf{Ax} + \mathbf{By} \leq \mathbf{b}, \\ &\mathbf{x} \in \mathbb{Z}^n, \\ &\mathbf{y} \in \mathbb{R}^p, \end{aligned}$$

where $\mathbf{b}$ is a $m \times 1$ vector and $\mathbf{A}$ and $\mathbf{B}$ are $m \times n$ and $m \times p$ matrices, respectively, which define the constraints of the problem. Notice that when $p = 0$, the problem becomes a pure integer programming problem, while $p \geq 1$ results in a mixed integer programming problem. Figure 4.7 is inspired by Junger et al. (2009) and adequately demonstrates the feasible region for a MILP in two dimensions, where $X_1 \in \mathbb{Z}$ and $X_2 \in \mathbb{R}$.

Mixed integer programming opens up a realm of possibilities in the mathematical modelling space and allows the modeller to make certain alterations to problems or their constraints in order to formulate the problem more effectively. Smith and Taskin (2007) list a few "tricks" that assist in reformulating linear programming problems by incorporating integer decision variables, some of which are discussed next.

Firstly, mixed integer programming allows the incorporation of "if-then-else" statements into a mathematical model. Consider, for example, a problem where $x \leq M$ and $y \leq N$, with $x, y \geq 0$. Suppose that if $x > Q_1$, with $Q_1 < M$, then y must be less than Q_2, with $Q_2 < N$. To enforce this condition, introduce a binary variable $z \in \{0, 1\}$ into the model such that $z = 1$ if $x > Q_1$ and zero elsewhere. Instead of writing $0 \leq x \leq M$, the constraint can now be written as $0 \leq x \leq Q_1 + z(M - Q_1)$. Similarly, the constraint $0 \leq y \leq N$ can be replaced with $0 \leq y \leq N - z(N - Q_2)$. Notice that when $z = 1$, x will be greater than Q_1 and y will be less than Q_2. Alternatively, if $z = 0$, it implies that x is less than or equal to Q_1 and no restriction is imposed on y. A great example of the aforementioned "if-then-else" logic is the widely published fixed-charge problem. In summary, the problem states that a factory can choose to engage in n activities, where the continuous variable x_i, $i = 1, \ldots, n$, represents the level of each activity. A variable cost vc_i is associated with each level of activity i that is undertaken. Additionally, every activity also incurs a fixed cost fc_i which remains the same regardless of the level of x_i. However, this fixed cost is equal to zero if activity i is discarded. The aim of the factory is to minimise the joint costs associated with all of its activities. The problem can then be formulated as follows:

$$\min \sum_i vc_i x_i + fc_i z_i, \tag{4.9}$$

subject to

$$\mathbf{Ax} \geq \mathbf{b}, \tag{4.10}$$

$$x_i \geq m_i z_i \quad \text{for} \quad i = 1, \ldots, n, \tag{4.11}$$

$$x_i \leq M_i z_i \quad \text{for} \quad i = 1, \ldots, n, \tag{4.12}$$

$$z_i \in \{0, 1\} \quad \text{for} \quad i = 1, \ldots, n. \tag{4.13}$$

In the fixed-charge problem, the variables M_i and m_i are specified to be upper and lower bounds for x_i, respectively, i.e. it is the highest and lowest levels of activity i which are allowed. If the factory can afford unlimited levels of activity i, then M_i can be set arbitrarily large (however, this is usually not the case, since activity levels are normally dependent on capacity and/or material constraints). Alternatively, it is clearly impossible to undertake negative levels of an activity, so m_i cannot be less than zero. Notice that when $z_i = 1$, it implies that a decision has been made to engage in activity i. With $z_i = 1$, the fixed costs fc_i are "switched on" and subsequently included in the objective function. However, if $z_i = 0$, the constraints in (4.11) and (4.12) will ensure that $x_i = 0$, which means that activity i is disregarded and that no fixed costs will be incurred. A detailed example of the fixed-charge problem is available in Section 12.1.1 of Brandimarte (2006).

Introducing integers into a programming problem can also assist with ensuring that at least

m out of M constraints are met. Suppose that a linear programming problem has M constraints of the form

$$\begin{aligned} \mathbf{a}_1\mathbf{x} &\leq b_1, \\ &\vdots \\ \mathbf{a}_M\mathbf{x} &\leq b_M. \end{aligned}$$

Assume that it is necessary that at least m of these constraints are met. The inequalities can then be reformulated as

$$\begin{aligned} \mathbf{a}_1\mathbf{x} &\leq b_1 + Nz_1, \\ &\vdots \\ \mathbf{a}_M\mathbf{x} &\leq b_M + Nz_M, \end{aligned}$$

where N is a sufficiently large number and $z_i \in \{0, 1\}$, $i = 1, \ldots, M$. The following constraint is then added:

$$\sum_{i=1}^{M} z_i = M - m.$$

The additional constraint suggest that if m constraints should hold, then there will be $M - m$ constraints which are not necessarily met (however, notice that they *can* be satisfied, since $\mathbf{a}_1\mathbf{x} \leq b_1$ implies that $\mathbf{a}_1\mathbf{x} \leq b_1 + N$). The above formulation shows that if any z_i is equal to one, the corresponding constraint in the original problem may not be met.

Lastly, binary integer variables can also be very helpful when transforming nonlinear functions (such as polynomials, $\log(x)$ functions and e^x functions) into linear functions via the process of linearisation. Brandimarte (2006) provides a detailed explanation of the piecewise linearisation of nonlinear functions, which is discussed next. While it is possible to employ nonlinear programming tools to model nonlinear dependencies between variables, the issue may be avoided by approximating a nonlinear function by means of linear splines via interpolation. Although commercial solvers can take quadratic problems as inputs, these problems remain much harder to solve and are computationally intensive. Quadratic optimisation problems become especially difficult for largescale problems in many dimensions, which is often the case with real-world problems that have thousands of data points. Ultimately, most robust commercial tools used for exact mathematical modelling are only able to handle linear programs effectively, as they are easier to solve and consume less time. To explain the concept of linearisation, consider a function in a one-dimensional space where the x-axis is divided into three sections by two grid

values, namely $x^{(1)}$ and $x^{(2)}$, which can be written as

$$f(x) = \begin{cases} m_1 x & \text{for } 0 \leq x \leq x^{(1)}, \\ m_2(x - x^{(1)}) + m_1 x^{(1)} & \text{for } x^{(1)} \leq x \leq x^{(2)}, \\ m_3(x - x^{(2)}) + m_2(x^{(2)} - x^{(1)}) + m_1 x^{(1)} & \text{for } x^{(2)} \leq x \leq x^{(3)}, \end{cases}$$

where $x^{(0)} = 0$ and $x^{(3)}$ is the last grid-value beyond which the function is not considered. If $m_1 \leq m_2 \leq m_3$, then $f(x)$ is an increasing function of x and is convex. Alternatively, if $m_1 \geq m_2 \geq m_3$ it can be seen that $f(x)$ is a decreasing function of x and is concave. Lastly, if the slopes m_i are arbitrary, the function is neither convex nor concave. Figure 4.8 below provides a graphical illustration of these three cases (similar to the examples presented by Brandimarte (2006) in Section 12.1 of the author's work).

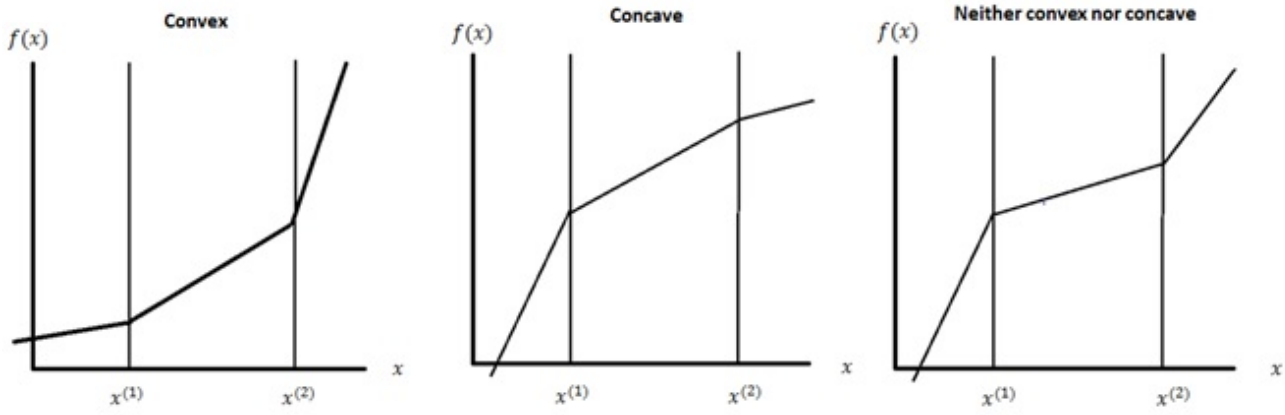


Figure 4.8: Linearisation of functions that are convex, concave and neither convex nor concave

In convex form, the function can easily be converted to a system of linear piecewise functions by creating three auxiliary variables y_1, y_2 and y_3 and substituting to obtain

$$\begin{aligned} x &= y_1 + y_2 + y_3, \\ 0 &\leq y_1 \leq x^{(1)}, \\ 0 &\leq y_2 \leq (x^{(2)} - x^{(1)}), \\ 0 &\leq y_3 \leq (x^{(3)} - x^{(2)}), \end{aligned}$$

which means that the function can be expressed as

$$f(x) = m_1 y_1 + m_2 y_2 + m_3 y_3.$$

In the optimal solution y_2 is only positive if y_1 has reached its upper bound, since $m_1 \leq m_2$. In

the same sense, y_3 is only greater than zero if both y_1 and y_2 are saturated. A similar notation can be followed for concave functions. Given this brief background, assume, in general, that a function is described by a set of knots (x_i, y_i) with $y_i = f(x_i)$ and $i = 1, \ldots, k$. In the examples provided in Figure 4.8, each function had $k = 4$ grid values on the x-axis and the same number of corresponding y-values can be obtained by evaluating $f(x)$ at each grid value (notice that only two grid values are shown in Figure 4.8, the other two grid values can be any two arbitrary points chosen at the opposite ends of each function, such as $x^{(0)} = 0$ and $x^{(3)} = M$, where $M > x^{(2)}$). Any point on the line between (x_i, y_i) and (x_{i+1}, y_{i+1}) can then be expressed as a convex combination in the following manner:

$$x = \lambda x_i + (1 - \lambda)x_{i+1},$$
$$y = \lambda y_1 + (1 - \lambda)y_{i+1},$$

where $0 \leq \lambda \leq 1$. Subsequently, a convex combination of the knots can be formed by

$$\mathbf{x} = \sum_{i=1}^{k} \lambda_i x_i,$$
$$\mathbf{y} = \sum_{i=1}^{k} \lambda_i y_i,$$

where

$$\sum_{i=1}^{k} \lambda_i = 1,\ \lambda_i \geq 0,$$

which produces the convex hull of the k knots. Since the function is convex and constantly increasing, it will always result in two adjacent λ_i weights being greater than or equal to zero, while the rest of the λ_i's will be exactly zero. The same applies for a concave function.

In a situation where the function is neither convex nor concave, additional tinkering with the linear interpolation approach is required to allow at most two adjacent λ_i coefficients to be positive. This is achieved by introducing $k - 1$ additional binary decision variables $s_i \in \{0, 1\}$, $i = 1, \ldots, k - 1$, into the formulation. The introduction of the aforementioned binary variables will subsequently transform any quadratic problem into a MILP. The purpose of the s-variables will be to force two neighbouring λ_i weights to be greater than or equal to zero. This is achieved by adding constraints (4.14)–(4.19) to the model, as demonstrated below:

$$x = \lambda x_i + (1 - \lambda)x_{i+1},$$
$$y = \lambda y_i + (1 - \lambda)y_{i+1},$$

subject to

$$0 \leq \lambda_1 \leq s_1, \tag{4.14}$$

$$0 \leq \lambda_2 \leq s_1 + s_2, \tag{4.15}$$

$$\vdots$$

$$0 \leq \lambda_{k-1} \leq s_{k-2} + s_{k-1}, \tag{4.16}$$

$$0 \leq \lambda_k \leq s_{k-1}, \tag{4.17}$$

$$\sum_{i=1}^{k} s_i = 1, \tag{4.18}$$

$$s_i \in \{0, 1\}. \tag{4.19}$$

To illustrate the need for the additional binary variables, consider the function in Figure 4.9 which is neither convex nor concave. The convex hull of the function is depicted by the grey shaded area.

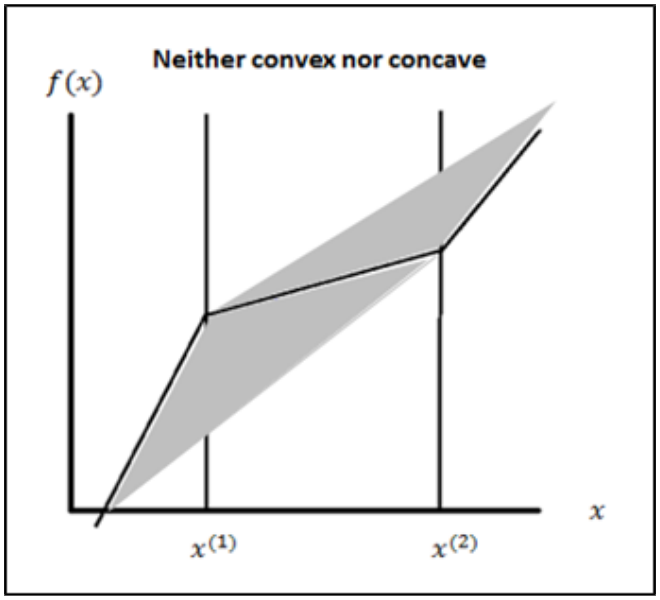


Figure 4.9: Linearisation of a function that is neither convex nor concave

If the binary s_i variables were not incorporated during the linearisation of the function in Figure 4.9, any point that is not equal to one of the grid values would lie somewhere in the convex hull, but not necessarily on one of the splines. As a result, the two λ_i weights that express the point as a convex combination of two grid values may not be automatically located next to one another. If any point is not located directly on a spline, it defeats the purpose of approximating the nonlinear function by means of a piecewise linear one, which will ultimately be an unsuccessful endeavour.

Linearisation is an important concept to grasp and will prove to be of great help in the suggested logistic regression approach presented in Chapter 5.

Given the general form of mixed integer programming problems and their possible applications, one of the most popular methods for solving MIPs – with a specific focus on MILPs – will be discussed next.

4.2.2 Solving mixed integer programming problems: the branch-and-bound method

The branch-and-bound method for solving mixed integer linear programming problems was first conceptualised by Land and Doig (1960). The method follows a strategy of "divide and conquer" by partitioning the feasible region into smaller subdivisions. In each subdivision, the algorithm solves the problem as if it was an LP by relaxing some of the integrality constraints. After each iteration, the algorithm evaluates the objective function value in order to establish upper and lower bounds for the optimal solution (these bounds will be explained shortly). The branch-and-bound method will then subsequently use this information to decide on which variable to "branch" next, at which point the process is repeated. Firstly, consider the definition of a relaxed problem, which is given below:

Definition 4.2.1. Let $P(Z) = \min_{\mathbf{x} \in Z} f(\mathbf{x})$ be a general minimisation problem where $Z \subseteq \mathbb{Z}^n$. The problem $P(S) = \min_{\mathbf{x} \in S} h(\mathbf{x})$, where $S \subseteq \mathbb{R}^n$, is then a relaxation of $P(Z)$ if $Z \subseteq S$ and $h(\mathbf{x}) \leq f(\mathbf{x}) \ \forall \ \mathbf{x} \in S$. The notation can easily be extended to a maximisation problem by swapping the inequality.

Given the definition of a relaxed problem, the systematic workings of the branch-and-bound algorithm will now be explained by summarising the information given by Gurobi Optimization (2017). The algorithm starts by solving a relaxed version of the MILP whereby all integer constraints are ignored. The relaxed problem therefore becomes a straightforward LP, which can be solved using approaches such as the simplex method. The original MILP will be referred to as P_0. Notice that if the problem is a maximisation problem, the optimal solution found by the LP will serve as an upper bound for the objective of P_0. Alternatively, if the problem is a minimisation problem, the LP solution is considered to be a lower bound for P_0. If the modeller is lucky enough such that the optimum solution of the first LP contains only integer values, it means that all integer constraints are satisfied and the branch-and-bound method will terminate. However, this is rarely the case in practice. Suppose that the decision variables x_i, $i \geq 1$ are required to take on only integer values as specified by the MILP, but the relaxed LP produced a fractional value $x_i = x_i^*$, $x_i^* \in \mathbb{R}$, for each one of these variables when an optimal solution for P_0 was obtained. The algorithm will then select an x_i to "branch" on whereby

two MILP subproblems, P_1 and P_2, are created. The first subproblem, P_1, will create a branch by adding the constraint $x_i \leq \lfloor x_i^* \rfloor$ while P_2 will branch on[2] $x_i \geq \lceil x_i^* \rceil$. The same approach that was applied to P_0 is then followed by solving relaxed versions of the two subproblems, P_1 and P_2, as two LP programming problems, LP_1 and LP_2. By iteratively repeating this process, the branch-and-bound method creates what is known as a search tree. The algorithm does not branch further on a node in the tree if the relaxed problem in that particular node is solved such that all integrality constraints are met.

At any point during the execution of the branch-and-bound method, the best integer solution that has been found so far is known as the *incumbent*. If an integer solution is found at one of the subsequent nodes where the objective function value is better than the current incumbent, the incumbent will be updated. Alternatively, as branching progresses throughout the tree, the algorithm might decide to discard a node, which is known as *fathoming*. The algorithm may decide to fathom a node if its solution is either infeasible or if the objective of the relaxed LP is larger than the current incumbent (if it is a minimisation problem – the opposite is true for a maximisation problem). The incumbent will always serve as a valid upper bound on the optimum solution if the problem being dealt with is a minimisation problem (it serves as a lower bound for a maximisation problem). This means that the algorithm never has to accept a solution with an objective function value higher than the incumbent.

At each iteration, the algorithm also produces a lower bound known as the *best bound*. At any point in the search, the best bound is equal to the minimum of all the objective function values across all of the leaf nodes [3] in the tree so far (if it is a minimisation problem). The difference between the best bound and the incumbent is then known as the *optimality gap*. When this gap is equal to zero, the branch-and-bound algorithm will have produced an optimal solution. In most mathematical software packages, the optimality gap is defined as

$$gap = \frac{|best\ bound - incumbent|}{\epsilon + incumbent},$$

where ϵ is some very small value, such as $1e-10$ (Gurobi Optimization, 2017). The existence of the incumbent and the best bound has some very important computational implications, as it enables the algorithm to enumerate only some branches (instead of having to evaluate every

[2]Different software packages have different ways of choosing the most appropriate variable on which to branch next, but all of them will ultimately arrive at the same solution in the end. One method mentioned by Bradley et al. (1977) is to select the branch with the largest objective function value obtained from the relaxed LP for evaluation (if the problem is a maximisation problem). This is known as last-generated-first-analysed branch-and-bound. By using this method, the results obtained from the simplex method during each subdivision is stored in a list. Any new information is then entered at the top of the list. Results are then first extracted from the top of the list to take the data obtained from the most recent nodes into account. The algorithm will therefore consider each subsequent subdivision based on the most recent data that has already been stored by the computer.

[3]A leaf node is a node in the tree where the algorithm has not created any branches yet.

branch in the entire tree). It is only necessary for the tree to evaluate a branch to a point where the objective function value of the node is worse than the current best integer solution.

In order to demonstrate how the branch-and-bound method works, the following example from Chapter 9 in Bradley et al. (1977) is shown. Consider the maximisation problem

$$\max\ f(x_1, x_2) = 5x_1 + 8x_2,$$

subject to

$$\begin{aligned} &x_1 + x_2 \leq 6, \\ &5x_1 + 9x_2 \leq 45, \\ &x_1, x_2 \geq 0,\ \mathbf{x} \in \mathbb{Z}^2. \end{aligned}$$

Figure 4.10 visually conveys the approach followed in solving the example shown above, which is subsequently discussed in more detail.

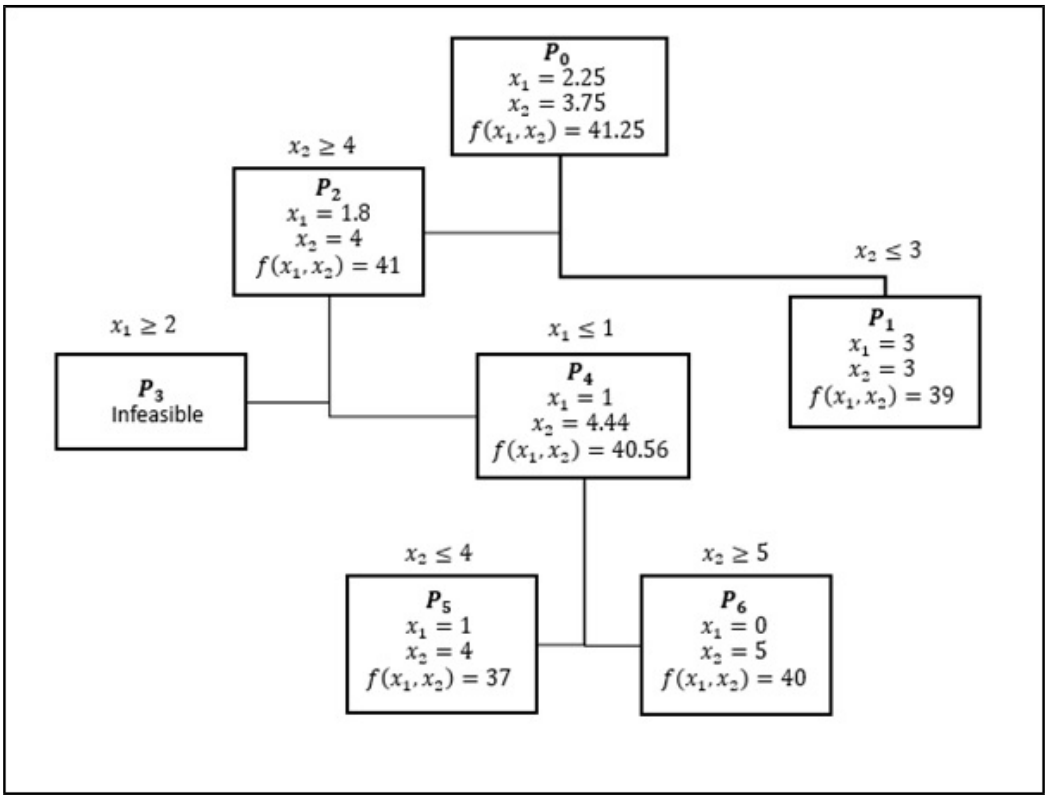


Figure 4.10: Branch-and-bound example

Solving a relaxed version of the MILP as an LP (this will be P_0) produces an objective value of $f(x_1, x_2) = 41.25$, with $x_1 = 2.25$ and $x_2 = 3.75$. The value 41.25 is seen as an upper bound for the objective value, noting that the MILP will never be able to achieve a better objective

function value. The branch-and-bound algorithm now has two choices: create two subproblems by branching on $x_1 \leq 2$ and $x_1 \geq 3$ or create two alternative subproblems by branching on $x_2 \leq 3$ and $x_2 \geq 4$. Suppose that x_2 is chosen and the algorithm proceeds by creating two subproblems, P_1 and P_2. Both problems will have exactly the same objective function to solve for and the same set of constraints as in P_0, with the difference being that P_1 has the additional constraint $x_2 \leq 3$ while P_2 has an extra constraint stating that $x_2 \geq 4$. Suppose that P_2 is solved first – again, as an LP where the integrality constraints of x_1 and x_2 are ignored. P_2 obtains an objective value of 41 by setting $x_2 = 4$ and $x_1 = 1.8$. Notice that the algorithm has not branched on P_2 yet, therefore, the best bound becomes 41.

The problem P_2 is then subdivided again into P_3 and P_4 due to x_1 not being an integer, with P_3 adding the constraint $x_1 \geq 2$ and P_4 adding $x_1 \leq 1$. Subproblem P_3 is infeasible, as the second constraint in P_0 cannot be met. Therefore, the branches stemming from P_3 are henceforth ignored and do not need to be evaluated. Notice that subproblems P_1 and P_4 now need to be considered. Suppose that P_4 is chosen. Solving a relaxed version of P_4 produces an optimal solution value of 40.56, with $x_1 = 1$ and $x_2 = 4.44$. Since the algorithm has not branched on P_4 yet, it can be considered for the best bound. However, the objective function value of 40.56 is less than the current best bound of 41 and so it remains unchanged.

It can be seen that x_2 is not an integer and so two subproblems, P_5 and P_6, originating from P_4 are created. Subproblem P_5 has $x_2 \leq 4$ and P_6 will add the constraint $x_2 \geq 5$. The branch-and-bound method now has problems P_1, P_5 and P_6 to consider. Solving P_5 as an LP results in $x_1 = 1$, $x_2 = 4$ and an objective function value of $f(x_1, x_2) = 37$. Since all solutions in P_5 are integer, the algorithm is not required to branch any further. The value of the incumbent is now set equal to 37, as it is the best integer solution that has been obtained so far. At this point, the optimality gap is equal to 10%, meaning that the modeller may opt to stop the branch-and-bound algorithm if he is satisfied with a solution that is at a distance of 10% from the optimal value. However, P_1 and P_6 may result in better objective function values, so suppose that P_6 is considered next. Evaluating P_6 yields $x_1 = 0$ and $x_2 = 5$, with an objective function value of 40. Again, the algorithm is not required to branch any further beyond P_6, as an integer solution has been obtained. The objective function value of 40 is better than the current incumbent of 37, so the incumbent is updated. At this point, the optimality gap is sitting at a narrow 2.5%. Again, the user can stop the algorithm at this point and use the solutions produced by P_6.

Alternatively, suppose that model is allowed to continue with P_1. Solving the relaxed version of P_1 yields an integer solution of $x_1 = 3$ and $x_2 = 3$, with $f(x_1, x_2) = 39$. Notice that if the best bound is taken as the maximum of the current leaf nodes in the tree, which are P_3, P_5, P_6 and P_1, then it will be set equal to 40, which coincides with P_6, as the objective function values associated with P_5 and P_1 are smaller and P_3 is infeasible.

The branch-and-bound algorithm can now terminate, as the incumbent and the best bound

is equal to one another, with an optimality gap of zero. The coordinates associated with P_6 are subsequently taken as the optimum solution to the integer programming problem. Indeed, the algorithm is not required to branch on P_1 for two reasons: 1) the solution produced by P_1 is an integer solution and 2) the objective value yielded by P_1 is less than the value produced by P_6. Even if the solution obtained by P_1 is not integer, the model is still not required to branch any further on P_1, unless its objective function value is greater than the $f(x_1, x_2) = 40$ yielded by P_6.

4.2.3 Algorithmic advances in solving mixed integer programming problems

Integer programming problems are NP problems [4](Papadimitriou, 1981) and are generally known to be difficult to solve, even by modern software packages. Since MIPs are a special case of IPs, they are considered to be at least as hard as IP problems. However, many believe MIPs to be harder. One way to reduce the execution time required to solve an integer programming problem (which will be explained shortly) is to tighten some of the inequalities. However, the tightening of inequalities become increasingly difficult when continuous variables are introduced into the integer problem. Many authors, such as Bayen et al. (2003), believe that practical MILPs, such as the aircraft scheduling problem, are NP-hard[5]. Recently, Del Pia et al. (2017) have shown that MIQP problems are NP-complete[6]. In light of the authors' work, many believe MILP problems to be NP-complete as well.

The field of mixed integer programming has enjoyed wide-spread attention over recent years, which has led to various noticeable improvements being applied to the solving of such problems. Gurobi Optimization (2017) provides a brief discussion on some of these advances, which will be summarised next.

First, the process of presolve is explained. The presolving of MIP problems is motivated by a pursuit to reduce the size of the problem by tightening its formulation and eliminating unnecessary constraints – if needed – before the branch-and-bound algorithm commences. To convey the idea of presolve, consider a very basic example of a problem that contains the following subset of constraints:

[4]In summary, NP problems are optimisation problems where it is theoretically possible to check the correctness of a solution in polynomial time, but it is very difficult to arrive at the solution itself. Even though it remains unproven, many researchers believe that it is not possible to solve NP problems in polynomial time.

[5]NP-hard problems are problems that are believed to be at least as hard as the hardest problems in NP, but they are not required to be in NP.

[6]NP-complete problems are considered to be the hardest problems in NP.

$$
\begin{aligned}
& x_1 + x_2 + x_3 = D, \\
& x_1 \leq a, \\
& x_2 \leq b, \\
& x_3 \leq c, \\
& x_1, x_2, x_3 \in \mathbb{R}.
\end{aligned}
$$

Furthermore, suppose that $D = a + b + c$ and $a \neq b \neq c$. Clearly, the inequalities will only be satisfied if $x_1 = a$, $x_2 = b$ and $x_3 = c$. The size of the problem can subsequently be reduced by eliminating these constraints and simply setting the three decision variables equal to the scalar values on the right-hand side of the inequalities. This is referred to as an LP-reduction, since no integrality constraints are specified. A MIP-reduction can be explained by considering, once more, a very basic example where the branch-and-bound method needs to solve a MIP that includes the following subset of constraints in its constraint matrix:

$$
\begin{aligned}
& 2ax_1 + 2ax_2 \leq a, \\
& x_1, x_2 \in \{0, 1\}.
\end{aligned}
$$

Simplifying the first constraint leads to the inequality $x_1 + x_2 \leq 0.5$. This implies that the constraint can only be satisfied when $x_1 = 0$ and $x_2 = 0$. The problem can therefore be reduced in size by discarding this constraint and setting the two variables equal to zero. In most cases, presolve results in a smaller feasible region for LPs (such as in the first example), but does not necessarily reduce the size of the feasible set for MIPs (as in the second example). This is an important consideration, since the objective function values obtained from the relaxed LPs in the branch-and-bound method serves as upper (if the problem is a maximisation problem) or lower (if the problem is a minimisation problem) bounds for all subsequent branches. Ultimately, tighter bounds should lead to more nodes in the tree being fathomed, resulting in less computational expense. However, the reduction of MIPs generally produces much more gains than the reduction of LPs, even though the former is a bit more difficult to achieve. Many commercial software packages, such as Gurobi Optimization and IBM's ILOG CPLEX, are able to automatically handle the process of presolving.

Next, the concept of *cutting planes* is introduced. Cutting plane algorithms were first conceived by Gomory (1963), but did not gain much momentum until significant developments were made in polyhedron theory. In summary, when the LP-relaxation in the branch-and-bound method produces a solution that is not feasible with respect to the MIP (i.e. it does not meet integrality constraints), the cutting plane approach "cuts" the infeasible solution from the poly-

hedron (feasible region) by introducing additional constraints to problem. By doing so, the problem is tightened due to the feasible region of the relaxed LP being reduced such that it is closer to the feasible region of the integer problem. These extra constraints are then considered by all subsequent nodes that are created by the branch-and-bound tree. To visually demonstrate the idea of a cutting plane, consider an integer problem in two dimensions as depicted in Figure 4.11. In Figure 4.11, the polyhedron pertaining to the integer problem is defined by the dashed lines, where the black coordinates represent possible integer feasible solutions. The feasible region of the relaxed LP is embodied by the solid black lines, where the red coordinate represents an optimum solution to the LP. However, the LP solution is not a feasible integer solution. A possible cutting plane would then be the inequality represented by the dotted black line, which "cuts" the aforementioned LP solution from all subsequent considerations.

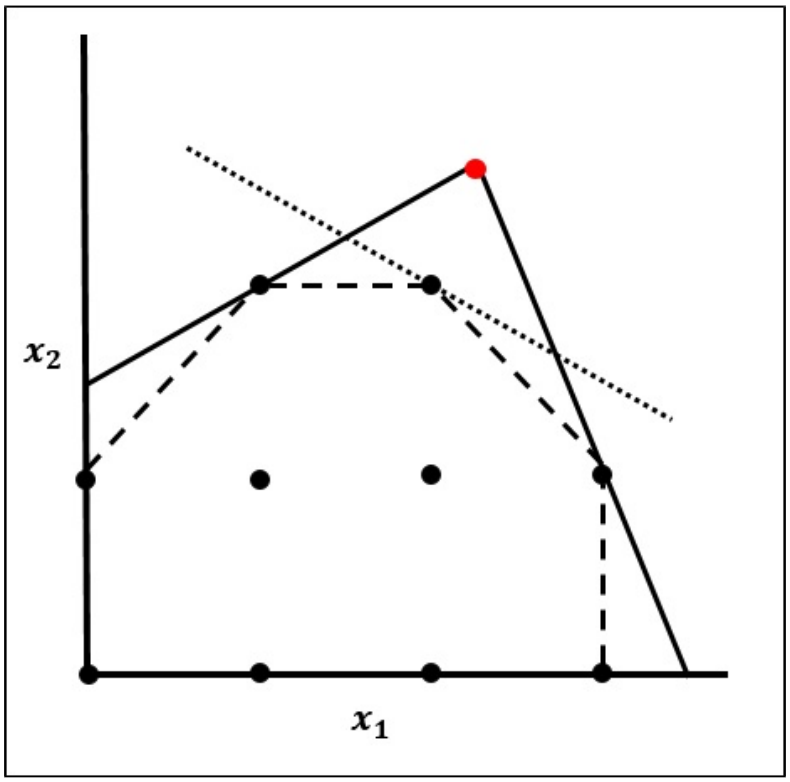


Figure 4.11: Cutting planes example

Alternatively, consider the following example given by Gurobi Optimization (2017) where a mixed integer programming problem has the following constraints:

$$c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 \leq D,$$
$$x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}.$$

Furthermore, suppose that $c_1 + c_2 + c_3 > D$ and that an optimum solution to the relaxed LP is

$x_5 = 1$, $x_4 = 0$ and $x_1 = x_2 = x_3 = a$ where a is some fractional value such that $a < 1$. In the optimal integer solution, the three decision variables x_1, x_2 and x_3 cannot all be equal to one, since $c_1 + c_2 + c_3$ is greater than D. Therefore, the inequality $x_1 + x_2 + x_3 \leq 2$ can be added as a constraint to the problem which "cuts" off the solution produced by the LP, thereby reducing the feasible region of the LP and forming a cutting plane. Specifically, Chvatal-Gomory cutting planes – which are a popular cutting plane method – can be expressed as follows:

Definition 4.2.2. Recall the general form of a MIP, which is given by

$$\max \ f(\mathbf{x}) = \mathbf{c}^T\mathbf{x},$$

subject to

$$\mathbf{A}\mathbf{x} \leq \mathbf{b},$$
$$\mathbf{x} \in \mathbb{Z}^n,$$

where $\mathbf{b}$ is a $m \times 1$ vector and $\mathbf{A}$ is a $m \times n$ matrix of constraints. If there exists a nonempty vector $\mathbf{d} \in \mathbb{R}^n$, $d_i \geq 0$, such that $\mathbf{d}^T\mathbf{A} \in \mathbb{Z}^n$ then the constraints $\mathbf{d}^T\mathbf{A}\mathbf{x} \leq \lceil \mathbf{d}^T\mathbf{b} \rceil$ are satisfied by all solutions for $f(\mathbf{x})$.

Cutting planes and branch-and-cut algorithms constitute an extensive field of study which encompasses many courses at tertiary institutions and vast works of literature that have been written on the subject. Mitchell (2000) provides a brief discussion on various aspects addressing the implementation of cutting planes and popular cutting plane methods that exist in literature. Solvers like Gurobi Optimization and IBM ILOG CPLEX include cutting plane approaches in their arsenal of algorithms directed at solving MIPs.

Lastly, *parallelism* is briefly addressed. The concept of parallelism is not based on innovative mathematical proposals – such as presolve and cutting planes – but can rather be seen as a product of extensive advances that have been made in information technology over recent years that has led to a tremendous increase in computing power and speedup of algorithms. Most software packages are able to effectively exploit multiple computer cores, which, in turn, means that several nodes in branch-and-bound trees can be processed and evaluated simultaneously alongside one another (in parallel).

At this point, concepts on fitting logistic regression models, methods for performing variable selection within regression modelling frameworks and formulating mathematical programming problems should all be clear. In the next chapter, the nonlinear logistic regression model introduced in Chapter 2 is formulated as linear programming problem (LP). The resulting LP is then expanded to facilitate variable selection (specifically best subset selection) by adding additional integer constraints to the model formulation, thereby transforming it into a MILP.

Chapter 5

Best subset selection and MILPs: a suggested approach

5.1 Introduction

The model formulations in (3.11)–(3.12), (3.14)–(3.15) and (3.19)–(3.20) clearly show that similarities exist between the formulations of lasso and ridge regression models and the best subset selection problem of Miller (2002). By modifying the model given in (3.1)–(3.2) slightly, consider the general form of a linear regression model with variable selection, which is given by

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2, \tag{5.1}$$

subject to

$$\|\beta\|_{p2}^{p1} \leq q. \tag{5.2}$$

In the case of logistic regression, (5.1) can be replaced with $\max_{\beta} \log L(\beta)$, where $\log L(\beta)$ is the log-likelihood function given in (2.11) of Section 2.3 in Chapter 2. For the Lagrangian form of (5.1)–(5.2), $\min_{\beta} \ -\log L(\beta)$ can be used in the objective function instead.

Notice that (5.1)–(5.2) can adequately represent a regression model with best subset selection, ridge regression or lasso regression by simply altering the values of p_1 and p_2 in (5.2). When $p_1 = 2$ and $p_2 = 2$, a ridge regression model is fitted. Alternatively, setting both $p_1 = 1$ and $p_2 = 1$ enables the model to make use of the lasso penalty to facilitate variable selection. Lastly, keeping $p_1 = 1$, but specifying $p_2 = 0$ results in best subset selection, where exactly q out of p possible variables are selected for inclusion in the final model (note that p refers to the number of potential predictors in the design matrix $\mathbf{X}$ and has no relation to either p_1 or p_2).

Bertsimas et al. (2016) provides the Lagrangian form of (5.1)–(5.2) in the case of best subset

selection, where $p_1 = 1$ and $p_2 = 0$, which is given by

$$\min_{\beta} \ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{l=1}^{p} I(\beta_l \neq 0), \tag{5.3}$$

where (5.3) might be seen as a special case of the objective functions used in models that utilise nonconvex penalties, such as MCP and SCAD. However, the authors note that, due to nonconvexity, (5.3) is not equivalent to (5.1)–(5.2) with $p_1 = 1$ and $p_2 = 0$. When (5.1)–(5.2) is applied in best subset selection, the user has precise control over the exact number of variables that enter the model. In turn, no clear link exists between λ and q in (5.3).

Bertsimas et al. (2016) addresses the best subset selection problem via a modern optimisation lens by using (5.1)–(5.2), with $p_1 = 1$ and $p_2 = 0$, to develop a mixed integer nonlinear optimsation approach (MIO) for choosing q out of p input variables in a linear regression model with n observations and a continuous dependent variable. Additionally, a discrete extension of modern first order continuous optimisation methods that produce high quality feasible solutions are also obtained and used by the authors as warm starts in their MIO. Their resulting approach is a MIQP that provides a guarantee on the optimality of the solution, can accommodate side constraints on the regression coefficients and can be extended to problems with the least absolute deviation loss as objective function (median regression). The suggested MIQP formulation provides optimal solutions for problems with n in the 1000's and p in the 100's and near-optimal solutions for problems with n in the 100's and p in the 1000's. Noting that (5.1)–(5.2) with best subset selection is an NP-hard problem (Natarajan, 1995), the authors state that polynomial times do not exist for the best subset selection approach unless it can be proven that P=NP. Instead, they consider their proposed model to be computationally tractable if it can be solved within a reasonable amount of time that is appropriate for the level of complexity inherent to problem being addressed. Empirical results suggest that solutions are achieved in minutes. In Bertsimas and King (2016), the work performed in Bertsimas et al. (2016) is extended to produce much more comprehensive regression models. This includes conditional constraints that cater for multicollinearity present in the data, variable transformations, forcing certain variables into the model (based on prior knowledge), introducing selective sparsity for variables with a group sparsity structure and ensuring that no more than one variable from a subset of m variables, where $m < p$, is included in the model. Maldonado et al. (2014) direct their efforts towards support vector machines (SVMs), as opposed to traditional regression models where $\mathbf{Y} \in \mathbb{R}$, and suggest a model formulation similar to that of Bertsimas et al. (2016).

Consider the following MIO formulation from Bertsimas et al. (2016) which addresses best subset selection in linear regression:

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\beta\|_2^2, \tag{5.4}$$

subject to

$$-M_{lL}z_l \leq \beta_l \leq M_{lU}z_l \quad \text{for } l = 1, \ldots, p, \tag{5.5}$$

$$z_l \in \{0, 1\} \quad \text{for } l = 1, \ldots, p, \tag{5.6}$$

$$\sum_{l=1}^{p} z_l \leq q. \tag{5.7}$$

While the objective function in (5.4) is of little concern within the context of logistic regression, the cardinality constraints that are used to enforce variable selection through mixed integer programming are noted. In the above set of constraints, z_l is a binary decision variable corresponding to the l-th regression coefficient β_l. Clearly, if $z_l = 1$ then β_l is non-zero and included in the model. Alternatively, when $z_l = 0$, the first constraint ensures that β_l is omitted from the final solution. The third constraint subsequently ensures that no more than q variables can be chosen as inputs in the final regression model by forcing the remaining $p - q$ regression coefficients to be exactly equal to zero. The values M_{lL} and M_{lU} serve as lower and upper bounds for the size of the l-th regression coefficient. While the choice of M_{lL} and M_{lU} is subjective, Bertsimas et al. (2016) lists a variety of approaches that assist in finding suitable values for these bounds. For numerical stability one would like to choose M_{lL} and M_{lU} such that the differences $-M_{lL} + \beta_l^*$ and $M_{lU} - \beta_l^*$ are as small as possible, where β_l^* is the optimum value of the l-th regression coefficient.

In this chapter, the best subset selection model in (5.4)–(5.7) developed by Bertsimas et al. (2016) is applied to the logistic regression model formulation. By formulating a linearised approximation of the log-likelihood objective function[1], the MLE of the logistic regression model can be presented as a linear programming problem (LP) which can subsequently be solved by employing the well-known simplex method when cardinaly constraints are not considered. When cardinality constraints are added to facilitate best subset selection, the modified logistic regression model becomes a MILP. Existing approaches towards the linearised approximation of the log-likelihood objective function are discussed next.

5.2 Existing approaches

After noting that most commercial MIO software packages struggle to handle nonlinear objective functions effectively (as is the case with logistic regression with feature selection), Sato et al. (2015) suggests a piecewise linear approximation of the logistic regression model objective function, which allows the model to be formulated as an LP. Variable selection is facilitated via the same binary z-variables shown in (5.4)–(5.7). Kamiya et al. (2019) embraces the concepts put

[1]Refer to Section 4.2.1 of Chapter 4 for a detailed explanation on the concept of linearisation.

forth by Sato et al. (2015) – which involves the formulating of dichotomous regression models as MIO problems – by presenting a multinomial logistic regression model with best subset selection as a mixed integer optimisation problem. Both sets of authors indicate that their respective models exhibit favourable generalisation characteristics and are solved within a reasonable amount of time, along with results that rival L_1-regularisation (otherwise known as lasso) and stepwise regression methods. Sato et al. (2015) first presents the logistic regression model with variable selection as a mixed integer nonlinear optimisation problem[2], where $\mathbf{X}_i$ is a $p \times 1$ vector of inputs for the i-th observation:

$$\min_{\beta,z} \; 2\sum_{i=1}^{n} f(Y_i(\mathbf{X}_i^T\beta)) + F\left(\sum_{l=1}^{p} z_l + 1\right), \tag{5.8}$$

subject to

$$z_l = 0 \Rightarrow \beta_l = 0 \quad \text{for } l = 1, \ldots, p, \tag{5.9}$$

$$z_l \in \{0, 1\} \quad \text{for } l = 1, \ldots, p, \tag{5.10}$$

where

$$f(v) = \log(1 + \exp(-v)) \tag{5.11}$$

is the logistic loss function, which is obtained from the log-likelihood function. Since $-\sum_{i=1}^{n} f(Y_i(\mathbf{X}_i^T\beta))$ is equal to the log-likelihood, the authors choose to minimise $\sum_{i=1}^{n} f(Y_i(\mathbf{X}_i^T\beta))$ instead of maximising the negative logistic loss function, which produces the same result. This would explain why the log-likelihood in (2.11) is maximised while the objective function in (5.8) is minimised. The second half of (5.8) involving F serves as a penalty that limits the number of features included in the final model. When $F = 2$, the most optimal model is selected based on the AIC, whereas $F = \log(n)$ will result in a final model that was selected using the BIC. Instead of requiring the modeller to explicitly specify the number of predictors permitted in the final model, as is done with the constraint shown in (5.7), the aforementioned penalty parameter allows the model to autmatically select the optimal number of inputs by choosing the classifier associated with the smallest AIC or BIC.

The constraint in (5.9) can be interpreted as a SOS1 (type one special ordered set), implying that only one element in the set can be non-zero. By rewriting (5.9) as the set

$$\{1 - z_l, \beta_l\} \quad \text{for } l = 1, \ldots, p,$$

the model is forced to set β_l equal to zero if $z_l = 0$. The same effect is achieved by the contraint in (5.5) in the model formulation presented in (5.4)–(5.7).

[2]Note that Sato et al. (2015) uses $\mathbf{Y} \in \{-1, 1\}$ and not $\mathbf{Y} \in \{0, 1\}$.

The authors proceed to formulate a piecewise linear approximation of the logistic regression model that finds the minimum of the objective function in (5.8) by a series of linear functions which "cut" away the non-feasible region located below the logistic loss function line. Specifically, given a set of k symmetric grid values $g_j, j = 1, \ldots, k$, which are not necessarily equally spaced and where the distance $(g_{j+1} - g_j)$ is not necessarily equal, the function in (5.11) may be approximated by the pointwise maximum of a set of linear inequalities, or

$$\begin{aligned} f(v) &= \max\{f'(g_j)(v - g_j) + f(g_j)\} \quad \text{for } j = 1, \ldots, k \\ &= \min\{t | t \geq f'(g_j)(v - g_j) + f(g_j)\} \quad \text{for } j = 1, \ldots, k. \end{aligned}$$

The nonlinear model in (5.8)–(5.10) is then formulated as the following MILP:

$$\min_{\beta, z} \; 2\sum_{i=1}^{n} t_i + F\left(\sum_{l=1}^{p} z_l + 1\right), \tag{5.12}$$

subject to

$$t_i \geq f'(g_j)(Y_i(\mathbf{X}_i^T\beta) - g_j) + f(g_j) \quad \text{for } j = 1, \ldots, k;\ i = 1, \ldots, n, \tag{5.13}$$

$$z_l = 0 \Rightarrow \beta_l = 0 \quad \text{for } l = 1, \ldots, p, \tag{5.14}$$

$$z_l \in \{0, 1\} \quad \text{for } l = 1, \ldots, p. \tag{5.15}$$

Figures 5.1 and 5.2 illustrate graphically how the objective function is approximated with the piecewise linear model in (5.12)–(5.15). The solid line represents the logistic loss function evaluated by the nonlinear model in (5.8)–(5.10), while the dashed lines represent a linear approximation of the same function. When the number of grid values k is relatively small, as in Figure 5.1, a greater disparity exists between the objective function of the nonlinear model and its linearised counterpart. This difference is demonstrated by the shaded triangle region in the figure. Due to the constraints imposed by (5.13), the objective function in (5.12)–(5.15) is optimised at the intersection of the two tangent lines, indicated by the solid coordinate. When more linear inequalities are added to the model, as shown in Figure 5.2, the relative gap between the results achieved by the nonlinear model and the approximated linear regression formulation narrows. The area of the shaded triangle region is significantly reduced and the optimal objective function value is achieved at the second intersection, which is located slightly above the first.

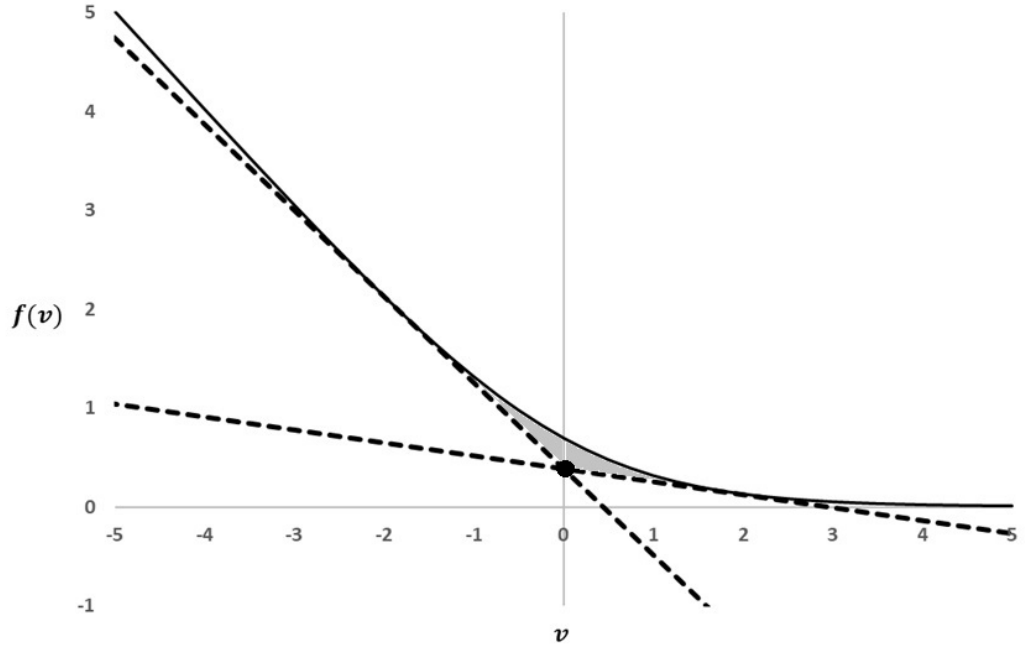


Figure 5.1: Linearised logistic regression model with 2 tangent lines

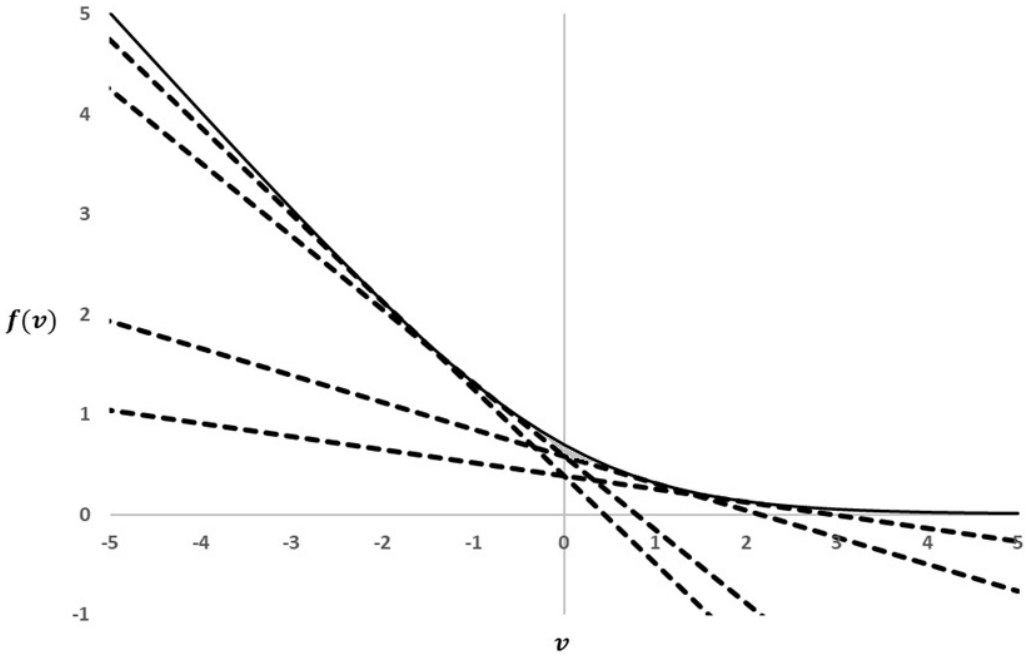


Figure 5.2: Linearised logistic regression model with 4 tangent lines

5.3 Suggested approach

An important attribute of the log-likelihood function is the fact that it is a concave function. As shown in Section 4.1.2 of Chapter 4, a local optimal solution for a convex function $f(x)$ will also be a globally optimal solution. Furthermore, if $f(x)$ is convex, then $-f(x)$ is concave, which implies that minimising $f(x)$ will yield an identical solution to the one obtained when $-f(x)$ is maximised. Due to the concave nature of the log-likelihood, a numerical approach taken towards the maximisation of the objective function in (2.11) would thus produce a globally optimal solution for the estimated coefficient vector $\hat{\beta}$.

Again, consider the log-likelihood objective function given in (2.11) in Chapter 2, which is expressed as

$$\log L(Y_1, \ldots, Y_n) = \sum_{i=1}^{n} Y_i(\beta^T \mathbf{X}_i) - \sum_{i=1}^{n} \log[1 + \exp(\beta^T \mathbf{X}_i)].$$

Notice that the first half of (2.11) is already linear in the predictor variables, which only requires the second half of the function relating to $\sum_{i=1}^{n} \log[1 + \exp(\beta^T \mathbf{X}_i)]$ to be linearised. The linearisation of the second part of (2.11) is facilitated through the use of a grid consisting of k equally spaced grid values g_j, $j = 1, \ldots k$, spanning the range $[A, B]$, where $A, B \in \mathbb{R}$ and the distance $(g_{j+1} - g_j)$ is identical for all j. For simplification, the choice of A and B used to define the range of the grid can be considered as arbitrary. However, suggestions on selecting a potentially appropriate grid are given in Chapter 6. A first attempt at producing a linearised version of the log-likelihood objective function is given by the following linear programming problem:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{k} \left(g_j Y_i - \log[1 + \exp(g_j)]\right) \lambda_{ij}, \tag{5.16}$$

subject to

$$\sum_{j=1}^{k} \lambda_{ij} g_j = \beta^T \mathbf{X}_i \quad \text{for } i = 1, \ldots, n, \tag{5.17}$$

$$\sum_{j=1}^{k} \lambda_{ij} = 1 \quad \text{for } i = 1, \ldots, n, \tag{5.18}$$

$$0 \leq \lambda_{ij} \leq 1 \quad \text{for } j = 1, \ldots, k;\ i = 1, \ldots, n, \tag{5.19}$$

where

- k is the number of grid values specified in the linearisation,
- g_j is the j-th grid value,
- λ_{ij} is the j-th weight value associated with the j-th grid value for the i-th observation.

Notice that each observation will have its own set of k weight values λ_{ij}, for $j = 1, \ldots, k$. In total, the constraint matrix contains $n(k+2)$ rows and $p + nk + 1$ columns, where the total number of variables p excludes the intercept term. This formulation now allows estimates for the regression parameter vector β, which optimise the log-likelihood function, to be obtained using exact approaches such as the simplex method. Consider the following proof which shows that the linearised model in (5.16)–(5.19) will yield an optimal solution for the log-likelihood in (2.11):

Theorem 5.1. For a small enough grid interval $(g_{j+1} - g_j)$, the linear programming problem (5.16)–(5.19) provides an optimal solution to the log-likelihood function

$$\log L(Y_1, \ldots, Y_n) = \sum_{i=1}^{n} Y_i(\beta^T \mathbf{X}_i) - \sum_{i=1}^{n} \log[1 + \exp(\beta^T \mathbf{X}_i)]$$

Proof. For $k = 2$, we have from constraint (5.17) that $\lambda_{i1} g_1 + \lambda_{i2} g_2 = \beta^T \mathbf{X}_i$, for $i = 1, \ldots, n$. For a specific case i, it can be shown that

$$\begin{aligned} Y_i(\beta^T \mathbf{X}_i) - \log[1 + \exp(\beta^T \mathbf{X}_i)] =& Y_i(\lambda_{i1} g_1 + \lambda_{i2} g_2) - \log[1 + \exp(\lambda_{i1} g_1 + \lambda_{i2} g_2)] \\ \leq& Y_i(\lambda_{i1} g_1 + \lambda_{i2} g_2) - (\lambda_{i1} \log[1 + \exp(g_1)] + \lambda_{i2} \log[1 + \exp(g_2)]) \end{aligned}$$

since $\lambda_{i1} g_1 + \lambda_{i2} g_2$ is a convex combination of the adjacent grid points g_1 and g_2, according to constraints (5.18) and (5.19). Therefore, for a large enough k, $g_{j+1} \approx g_j$ such that

$$\begin{aligned} Y_i(\beta^T \mathbf{X}_i) - \log[1 + \exp(\beta^T \mathbf{X}_i)] &\approx Y_i \left(\sum_{j=1}^{k} \lambda_{ij} g_j \right) - \sum_{j=1}^{k} \lambda_{ij} \log[1 + \exp(g_j)] \\ &= \sum_{j=1}^{k} (g_j Y_i - \log[1 + \exp(g_j)]) \lambda_{ij}. \end{aligned}$$

□

The absence of cardinality constraints in the LP given in (5.16)–(5.19) implies that a logistic regression model is fitted to the data without any variable selection. By applying the logic of Bertsimas et al. (2016) in (5.4)–(5.7) and Sato et al. (2015) in (5.12)–(5.15) to the model in (5.16)–(5.19), variable selection for logistic regression – specifically best subset selection – is introduced by the following model formulation below:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{k} (g_j Y_i - \log[1 + \exp(g_j)]) \lambda_{ij}, \tag{5.20}$$

subject to

$$\sum_{j=1}^{k} \lambda_{ij} g_j = \beta^T \mathbf{X}_i \quad \text{for } i = 1, \ldots, n, \tag{5.21}$$

$$\sum_{j=1}^{k} \lambda_{ij} = 1 \quad \text{for } i = 1, \ldots, n, \tag{5.22}$$

$$0 \leq \lambda_{ij} \leq 1 \quad \text{for } j = 1, \ldots, k;\ i = 1, \ldots, n, \tag{5.23}$$

$$-M_{lL} z_l \leq \beta_l \leq M_{lU} z_l \quad \text{for } l = 1, \ldots, p, \tag{5.24}$$

$$z_l \in \{0, 1\} \quad \text{for } l = 1, \ldots, p, \tag{5.25}$$

$$\sum_{l=1}^{p} z_l \leq q. \tag{5.26}$$

5.4 Comparison with existing approaches

The formulation of the linearised logistic regression model with best subset selection shown in (5.20)–(5.26) is premised on the same motivations listed by Sato et al. (2015), which mainly involves three critical concepts, namely:

- As is the case with the model presented by Bertsimas et al. (2016), Sato et al. (2015) makes use of binary z-variables to facilitate variable selection, which has proven to be useful when feature selection is carried out within a mixed integer optimisation framework.
- A linear approximation of the nonlinear logistic regression model objective function is proposed so as to allow the regression model with variable selection to be formulated as a MILP.
- The use of a mixed integer linear optimisation model is suggested so as to provide the end user with a guarantee on the optimality of the solution, as opposed to more traditional variable selection methods that dependend on the evaluation of p-values and variable significance levels[3].

Indeed, the model shown in (5.20)–(5.26) and MILP proposed by Sato et al. (2015) share many similarities, especially considering that each observation, for $i = 1, \ldots, n$, is associated with k constraints in the constraint matrix. However, a key difference that exists between the two MILP formulations is the fact that the logistic regression model in (5.12)–(5.15) attempts

[3]Recall from Chapter 3 that stepwise regression approaches, such as forward and backward regression, will produce a single "good" model fit. However, these methods fail to provide the modeller with some sort of a gaurantee which shows that the resulting solution is the best possible model found amongst all combinations of predictors (Kutner et al., 2005).

to linearise the logistic loss function, whereas the model suggested in (5.20)–(5.26) attempts to directly linearise the log-likelihood function itself. Consider Figure 5.3 and the subsequent discussion thereof.

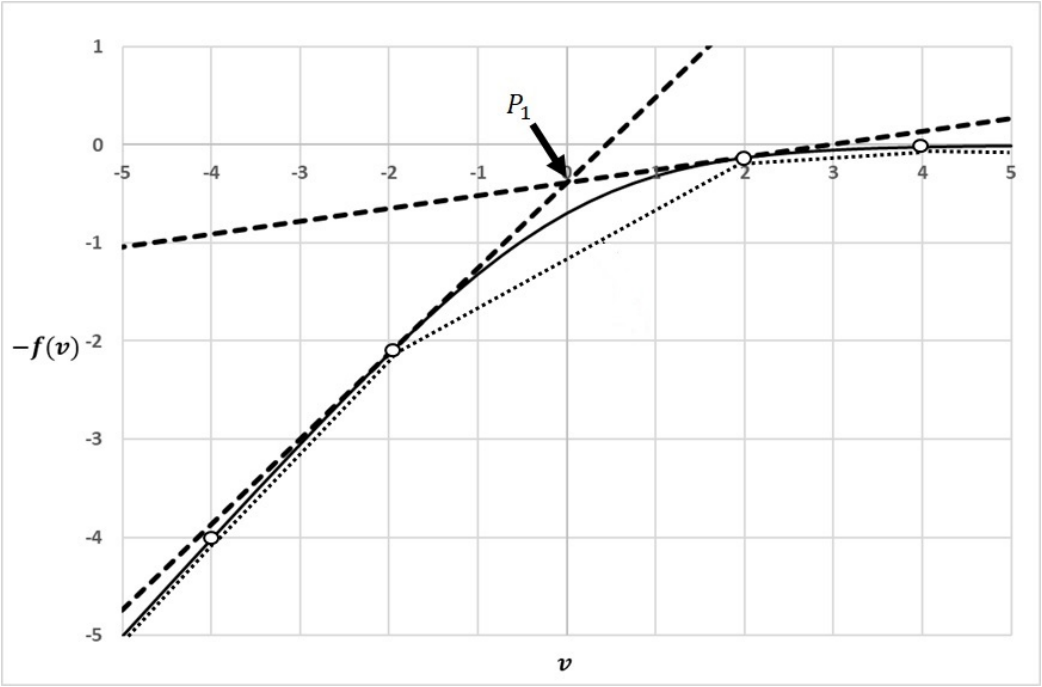


Figure 5.3: Comparison of two linearised logistic regression models

In Figure 5.3 the nonlinear log-likelihood function, which is equal to $-f(v)$ in (5.11), is plotted as a solid line. The two dashed lines represent the linear approximation of the logistic loss function as put forth by Sato et al. (2015) in (5.12)–(5.15), which was also shown previously in Figure 5.1. Alternatively, the dotted lines visually convey the linearisation of the log-likelihood function proposed in (5.20)–(5.26). If the number of grid values k are not substantially large, a larger gap will exist between the optimal objective function value achieved by both linearised models and the maximum log-likelihood value found by the nonlinear model. Specifically, the model in (5.12)–(5.15) will be optimised somewhere close to coordinate P_1, whereas the logistic regression model in (5.20)–(5.26) will yield an optimal log-likelihood value that is located on the dotted line. In fact, the maximised log-likelihood value for (5.20)–(5.26) will be a convex combination of two adjacent grid values (note that the grid values are shown as empty circle coordinates in Figure 5.3).

The following notable differences exist between the two linearised logistic regression models:

1. Let k_{max} be a significantly large number that tends to infinity. While $k \ll k_{max}$, the linearised model in (5.12)–(5.15) will tend to overestimate the log-likelihood function, whereas the linearised model in (5.20)–(5.26) will tend to underestimate it.

2. Sato et al. (2015) does not explicitly specify a cardinality parameter to facilitate variable selection, as is the case with constraint (5.26) in the model in (5.20)–(5.26) and constraint (5.7) in the model proposed by Bertsimas et al. (2016). Instead, the authors include a penalty term in the objective function, similar to the lagrangian form of lasso.

3. A different approach is followed for selecting the appropriate grid values g_j, $j = 1, \ldots, k$, that are utilised in the linearisation performed by the two respective models.

Each of the points listed above will now be addressed. Firstly, due to the approach followed during linearisation, the optimal objective function value obtained from the model proposed by Sato et al. (2015) will serve as an upper bound for the nonlinear log-likelihood in (2.11). Specifically, the authors state that their linearised approximation will act as an underestimator for the logistic loss function. In turn, if the model in (5.12)–(5.15) underestimates $f(v)$, it will overestimate $-f(v)$. Given that $-\sum_{i=1}^{n} f(Y_i(\mathbf{X}_i^T\beta)) = \log L(Y_1, \ldots, Y_n)$, it would imply that the log-likelihood is overestimated. It should be noted, however, that the authors' approach towards linearisation is motivated by their use of the AIC criterion for feature selection. With reference to (3.3), recall that the AIC is expressed as $-2\log L + 2q$, where q is the number of estimated parameters. A model with a smaller AIC value is preferred. It can therefore be seen why the authors choose to use the logistic loss function in their approximation, due to the inclusion of $-\log L$ in the AIC calculation. Alternatively, the solution found by the suggested model in (5.20)–(5.26) will serve as a lower bound for the log-likelihood MLE. Simply put, the proposed model in (5.20)–(5.26) yields a more conservative estimate and provides assurance that model fit can only be improved. This can perhaps be considered as a positive attribute inherent to the suggested methodology outlined in (5.20)–(5.26), as opposed to an overly optimistic approach where the optimal value of the log-likelihood might be overstated.

With regards to the second point listed above, recall that the model in (5.12)–(5.15) does not accommodate the explicit specification of a cardinality parameter q, but instead uses a penalty term to automatically select the number of inputs in the final model. For this reason, it may be argued that the linearised model with variable selection in (5.12)–(5.15) is more akin to penalised regression methods such as lasso, ridge regression and the elastic net. While the penalised regression model in (5.12)–(5.15) may appear to be more attractive due to not having to specify a cardinality parameter q outright (as in best subset selection), caution should be exercised against allowing the model to autonomously regulate the final number of variables by itself. Empirical studies have shown that criteria such as the AIC or BIC might have a tendency to favour models with a larger number of variables (refer to Section 3.2 of Chapter 3), unless the size of the modelling set is sufficiently large (Kutner et al., 2005).

Finally, the grid of knot values utilised during linearisation will now be discussed in more detail. Sato et al. (2015) proposes a greedy algorithm that sequentially adds tangent lines one by

one to the model whereby each additional line attemps to reduce the area of the shaded triangle shown in Figures 5.1 and 5.2, thereby decreasing the gap that exists between the nonlinear model and its linearised version. Since the area of the triangle obtained in each step can be calculated by using the coordinates of its three corners, the greedy algorithm proceeds to add the line that results in the biggest reduction of the triangle's size. Specifcally, the authors utilise three sets of grid values in their experiments: $G_1 = \{\pm\infty, \pm 1.9, 0\}$, $G_2 = \{\pm\infty, \pm 3.55, \pm 1.9, \pm 0.89, 0\}$ and $G_3 = \{\pm\infty, \pm 5.16, \pm 3.55, \pm 2.63, \pm 1.9, \pm 1.37, \pm 0.89, \pm 0.44, 0\}$. Results indicate that the formulation in (5.12)–(5.15) often produced the most favourable models when G_3 was used, which can be expected, since a greater number of tangent lines leads to a smaller gap between the nonlinear and linear objective function values. Alternatively, the linearised model in (5.20)–(5.26) makes use of an entirely different set of grid values. The model does not add the coordinates of the knots in an iterative fashion, but instead requires the modeller to explicitly specify the beginning and end points of the grid, along with the number of grid values k that the user would like the model to employ during its execution. The linearised logistic regression model will then proceed to utilise a grid of k equally spaced knots that lie within the aforementioned predetermined range. It should also be noted that the coordinates in the grid itself would most likely differ greatly from the ones shown in G_1, G_2 and G_3 in most settings (refer to the results presented in Chapters 6 and 7, where grids between -20 and 20 or -30 and 30 were often utilised and where the number of grid values regularly involved $k \in \{401, 161, 81, 41\}$).

While each of the models make use of disparate sets of grid values and differ in their linearisation, added to their similarities is the fact that longer model execution times can be expected as the number of grid values k increases. This can be seen in Chapter 7 as well as in the empirical results shown by Sato et al. (2015) when G_3 is used instead of G_1 or G_2. A trade-off therefore exists between the accuracy of the final model[4] and the time required to find the optimal objective function value. However, it might be argued that the suggested model proposed in (5.20)–(5.26) allows for the specification of a grid of values in a more "user-friendly" manner. Consider the coordinates shown in G_1, G_2 and G_3 and notice that these values are not particularly intuitive if they need to be specified explicitly – especially for a novice modeller. One would therefore have to rely on the suggested greedy algorithm to determine these knot values automatically in what can almost be considered a "black-box" approach. Alternatively, the approach suggested in Chapter 6 provides the user with a relatively intuitive and easy-to-follow method for determining a suitable grid of coordinates. This approach is less mathematically intensive and has proven to work well when applied to the various models listed throughout Chapters 6 and 7. Ultimately, the modeller will have to determine his or her preference towards algorithmic transparency and ease of understanding versus his or her inclination towards model autonomy

[4]Observe that, for both linearised approaches, the gap between the nonlinear objective function value and the linearised approximation will decrease as more grid values are employed by the linearised model.

and speed of execution.

In the next chapter, the linearised logistic regression model without variable selection in (5.16)–(5.19) and with subset selection in (5.20)–(5.26) is fitted to both simulated and real-world datasets to demonstrate the application of the LP and subsequent MILP formulations.

Chapter 6

Computational results

In order to assess the performance of the proposed formulations presented in equations (5.16)–(5.19) and (5.20)–(5.26), the linearised logistic regression model with variable selection was implemented and solved using the commercial product IBM ILOG CPLEX Optimization studio, version 12.6. The results of these efforts were then compared to runs that were executed on the same datasets using SAS's PROC LOGISTIC procedure in a 64-bit version of SAS Enterprise Guide 7.1[1]. Note, however, that IBM ILOG CPLEX is a mathematical programming solver and not a statistical analysis tool.

Ultimately, obtaining results comparable to (or better than) those achieved in SAS would be ideal, since its PROC LOGISTIC procedure is considered to be the benchmark in this exercise. Additionally, all simulated datasets that were used for synthetic model runs were generated in SAS Enterprise Guide. All work was carried out on a desktop computer with Intel i7-3770 3.40GHz processor, 16 GB of RAM and 64-bit Windows 7 operating system.

The reader should note that, throughout the chapter, the term "CPLEX run" refers to the linearised formulation produced in equations (5.16)–(5.19) and (5.20)–(5.26), whereas the term "SAS run" refers to the same model carried out using SAS's logistic procedure. Lastly, it should be known that for Chapters 6 and 7, the quantities p and q *do not* include the intercept term of the logistic regression model. This means that a logistic regression problem that considers a total of p potential features will be based on a design matrix with $p + 1$ columns, where an additional column of 1's is added for the intercept term.

[1]Default settings were used in the SAS logistic procedure when binary regression models were fitted to the datasets presented in this chapter. Apart from explicitly stating the DESCENDING option in the PROC LOGISTIC statement – due to the fact that an event was coded as $Y = 1$ – no alternative options were specified.

6.1 Simulated data runs

To perform comparisons on simulated data, a design matrix $\mathbf{X}$ that contains p input variables from a standard normal distribution for a total of n observations, was generated. To obtain a binary response vector $\mathbf{Y} \in \{0, 1\}$, the following steps were carried out:

1. Generate a $n \times 1$ vector $\mathbf{U} \in [0, 1]$ that contains n uniformly distributed random variables.

2. Generate a $n \times 1$ vector $\mathbf{L}$ that will contain n random variables having a logistic distribution by utilising the vector $\mathbf{U}$ obtained in step 1. The i-th entry of $\mathbf{L}$ is given by $L_i = \log[U_i/(1-U_i)]$.

3. Generate a $n \times 1$ vector $\mathbf{Y}^c$ which represents a latent ouput variable with a logistic error distribution, where the i-th entry of $\mathbf{Y}^c$ is given by $Y_i^c = \sum_{l \in Q} \beta_l^* X_{il} + L_i$. In the aforementioned summation, $\mathbf{X}$ is the input variable matrix containing p variables from a standard normal distribution, $\beta_l^* = 1$ for $l \in Q$ and $\beta_j^* = 0$ for $j \notin Q$. β_l^* is the true underlying value specified for the l-th regression parameter.

4. To produce a binary response vector containing either 0 or 1 entries, an $n \times 1$ vector $\mathbf{Y}$ was generated, where $Y_i = 1$ if $Y_i^c \geq 0$, else $Y_i = 0$ if $Y_i^c < 0$.

Initially, a sample containing $n = 1\ 000$ observations with $p = 100$ standard normal random variables was generated. A column of 1's was concatenated horizontally to the matrix of input variables in order to compensate for the intercept of the regression model, meaning that the design matrix contains a total of $p + 1$ columns. A set of 10 equi-spaced variables were then chosen to influence the response and have true regression coefficients greater than zero. Given that X_0 represents the intercept term, the variables $X_1, X_{11}, X_{21}, X_{31}, X_{41}, X_{51}, X_{61}, X_{71}, X_{81}$ and X_{91} were all selected to be associated with a regression coefficient of one, i.e. $\beta_l^* = 1$ where $l \in Q$ and Q is the set $\{1, 11, 21, 31, 41, 51, 61, 71, 81, 91\}$. The response vector was then constructed as a linear combination of these 10 variables using steps 3 and 4 outlined above. The dataset contained a total of 500 "goods" or observations with $Y = 0$ and 500 "bads" or observations with $Y = 1$. Note that no variable selection was performed in either SAS or CPLEX during this simulation. The purpose is to firstly establish whether the proposed linearised formulation in (5.16)–(5.19) yields the desired results when used to fit a logistic regression model before cardinality constraints come into play.

The logistic regression model in SAS achieves a maximum log-likelihood of -273.836. The iterative algorithm employed by SAS appears to be quite successful in separating the estimated regression coefficients associated with the 10 variables mentioned above (that were set to influence the target variable) from the parameter estimates obtained for the other 90 variables. The fitted coefficient values for the entire collection of non-influential predictors are scattered

around the zero-line, with the largest deviations being close to 0.2 or -0.2. Alternatively, all parameter estimates that were found for the 10 chosen variables had the correct sign and were far-removed from the other fitted coefficients, with most estimates scattered between 1.4 and 1.5 (with the exception of the coefficient obtained for X_{11}). Overall, the results can be interpreted as desirable, given the clear separation between the parameter estimates for the two groups and their relative close proximity to their true underlying values. The fact that the model tends to overestimate the coefficients for the 10 variables that were selected to influence the response, is noted, with a difference of roughly 0.5 between the estimated and true underlying parameter values for the majority of these inputs. However, given the relatively small size of the sample, this can be expected to improve with an increase in the number of observations.

Using a grid of 401 knots that span between -20 and 20 with 0.1 intervals, a logistic regression model is fitted to the data using the linearised model in (5.16)–(5.19). The model produces identical results to those found in the SAS run, with a maximum log-likelihood of -273.9 achieved in 20.56 seconds, along with parameter estimates that are nearly indistinguishable from those obtained by SAS's iterative algorithm. Consequently, the same interpretation applies.

A second run was performed wherein the range of the grid used in the linearisation was kept the same (-20 and 20), but intervals were specified to have a length of 0.01, resulting in a total of 4 001 grid values (as opposed to the 401 knots used in the previous run). This was done in order to ascertain whether or not an improvement in the model can be seen by reducing the distance between the grid values (this is based on the notion – as mentioned in Chapter 5 – that as the distance between the knots tend to zero, the nonlinear log-likelihood function employed by the iterative algorithm is obtained). While the linearised model only produces results after 3 minutes and 24 seconds (roughly 3 minutes more than needed by the previous run where only 401 grid values are employed), barely no real increase in the log-likelihood is recorded, with a maximum objective function value of -273.836 obtained (an improvement of only 0.044 over the previous run). The fitted parameter estimates were nearly identical to those found in the previous run. The empirical studies presented in Chapter 7 examine the trade-off between model execution time and the accuracy of results when the number of grid values which are employed in the linearised approximation, is varied.

Figure 6.1 illustrates the regression coefficient estimates produced by the aforementioned SAS and CPLEX runs. The x-axis denotes the variable number, from 0 to 100, while the y-axis represents the corresponding value of the estimated regression parameter for the variable in question. Note how the fitted coefficients produced for the 10 variables that were selected are located far above the rest and tend to be closer to one, while the rest are situated closer to zero.

Runs were then executed where the number of observations contained in the dataset, n, were increased. Table 6.1 shows log-likelihood values obtained for different values of n and the subsequent comparison thereof to the MLE produced by SAS.

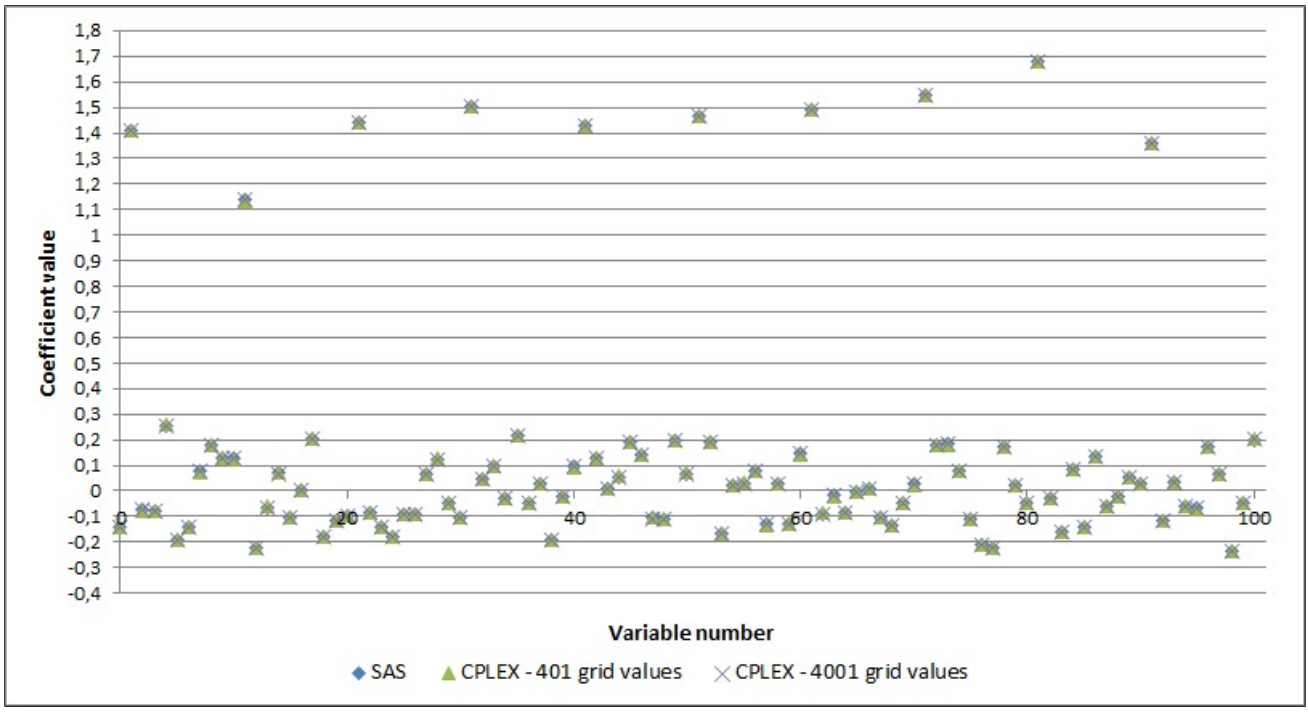


Figure 6.1: Regression parameter estimates obtained by SAS and CPLEX model runs

Table 6.1: Results of CPLEX and SAS runs

n	$\log L$: CPLEX linearised run	$\log L$: SAS logistic procedure
3 000	-957.671	-957.427
5 000	-1704.744	-1704.3
10 000	-3447.104	-3446.2
20 000	-7001.821	-6999.979
50 000	-17596.363	-17577.649

While not shown graphically, it should be noted that as the number of observations in the training set, n, increases, both the linearised model used in CPLEX and the iterative model employed by SAS obtain parameter estimates that are located increasingly closer to their true underlying values. As the number of records within the design matrix on which the classifier is being trained, becomes greater, it allows the model to produce a solution for the estimated coefficient vector that is a much more unbiased estimate of the true underlying parameter vector (Potts and Patetta, 1999). At $n = 20\ 000$ observations, the estimated coefficients for the 10 inputs that were chosen to influence the response (with true underlying parameter of $\beta^* = 1$) were close to 1, while the fitted parameters for the remaining 90 variables were all found within an extremely narrow range centered around the zero-line, with all values between -0.05 and 0.05. At $n = 50\ 000$, fitted coefficients for the selected 10 variables were almost exactly equal to one, while the parameter estimates for the remaining 90 variables were scattered close to the zero-

line. Notice from Table 6.1, however, that the log-likelihood values produced by the linearised model in CPLEX are marginally lower than the objective function values obtained in SAS. This can be attributed to the fact that the proposed model in (5.16)–(5.19) serves as a linear approximation of the log-likelihood function in (2.11). Recall that the model in (5.16)–(5.19) is as an underestimator for the nonlinear log-likelihood. The model in CPLEX will only yield maximum likelihood function values identical to those in SAS when k is substantially inflated and tends to infinity.

Overall, the proposed linearised model appears to fair well and yields log-likelihood objective function values that compares favourably with the nonlinear model in SAS. The parameter estimates obtained for the variables chosen to influence the response (with true underlying value of one) are forced to be much higher, while the fitted coefficients produced for the set of unselected inputs (with true underlying value of zero) are scattered with an increasingly narrow range about the zero-line. It should be noted, however, that these results are based on a design matrix with a very weak inter-variable correlation structure. The next section explores the behaviour of the proposed linearised MLE functon fitted to data with correlated outputs.

6.2 Simulated data runs with correlated inputs

For these simulations, the same approach outlined in Section 6.1 was utilised. However, a key difference to note is that linear correlations were deliberately introduced to create between-variable correlations within the design matrix. Specifically, correlated variables were generated such that the ij-th entry of the correlation matrix is given by $\rho^{|i-j|}$ for some specified correlation value ρ. This was done by using a variation of Cholesky's decomposition (Press et al., 1992) outlined in the following steps:

1. For some specified value ρ, generate a correlation matrix $\mathbf{\Gamma}$ such that the ij-th entry is given by $\Gamma_{ij} = \rho^{|i-j|}$.

2. Obtain a $p \times p$ diagonal matrix $\mathbf{D}$ where the entries along the diagonal are given by $D_{ii} = \sqrt{\mathbf{\Gamma}_{ii}}$.

3. Find a $p \times p$ matrix $\mathbf{R}$ by setting $\mathbf{R} = \mathbf{D}^{-1}\mathbf{\Gamma}\mathbf{D}^{-1}$.

4. Find a $p \times p$ matrix $\mathbf{C}$ where the entries of $\mathbf{C}$ are obtained as follows:

   a. $C_{ij} = R_{ij}/\sqrt{R_{ji}}$ if $j = 1$.

   b. $C_{ij} = 0$ if $j > i$.

   c. $C_{ij} = \sum_{k=1}^{j-1} (C_{ik}C_{jk}) \,/\, \left(\sum_{k=1}^{j-1} C_{jk}^2\right)$ for $j < i$ and $j \neq i$.

5. Obtain a new design matrix $\mathbf{X}_{new}$ by multiplying the current input variable matrix containing p standard normal random variables with the transpose of $\mathbf{C}$, or $\mathbf{X}_{new} = \mathbf{X}\mathbf{C}^T$.

The new design matrix obtained in step 5 will now have a correlation matrix closely resembling the one specified in step 1.

For the first correlations run, 500 observations and 100 variables from a standard normal distribution were generated. Once more, a column of 1's is added to the matrix of inputs to compensate for the intercept term of the regression model. By using the steps outlined above, correlations are introduced into the design matrix by specifying $\rho = 0.5$ as set out in step 1. Again, 10 equi-spaced variables are chosen to have an effect on the response by stipulating that the inputs $X_1, X_{11}, X_{21}, X_{31}, X_{41}, X_{51}, X_{61}, X_{71}, X_{81}$ and X_{91} all have a true regression coefficient of one, i.e. $\beta_l^* = 1$, $l \in Q$, where β_l^* is the true underlying value of the regression parameter. The binary dependent vector $\mathbf{Y}$ is obtained in the same way as before. The dataset contained a total of 244 "goods" or observations with $Y = 0$ and 246 "bads" or observations with $Y = 1$.

When $\rho = 0.5$, SAS obtains a maximum log-likelihood value of -107.643. The model fitted by the linearised logistic regression executed within CPLEX, on the other hand, attained a log-likelihood value of -108.328 after 10.38 seconds of execution by utilising a grid of 401 knots spanning the range [-20, 20] with intervals of 0.1. Additionally, the linearised model was refitted in CPLEX where a grid consisting of 4 001 knots was used – again with a range of [-20, 20], but with 0.01 intervals instead. As with previous simulations, this was done in order to gauge whether or not the linearised model produces more accurate results alongside an increase in the number grid values. However, after fitting the model for 1 minute and 51 seconds, an objective value for the likelihood function of -108.305 was obtained, resulting in a meager increase of 0.023, or 0.02%. In contrast, the time consumed by fitting the model increased nearly ten-fold by 969.36%. After noting that no substantial difference exists between the parameter estimates obtained by the two linearised model runs, it can be inferred – as in the previous section – that a substantial increase in the size of the grid in the linearised model does not necessarily amount to a meaningful improvement in accuracy.

When the parameter estimates themselves are analysed, it can be seen that very little difference exists between the fitted regression coefficients produced by the two respective models. It is noted that the coefficients obtained by both SAS and CPLEX deviate quite wildly from their true underlying parameter values. However, this can be expected in the presence of moderate to severe linear correlations that exist between input variables. On a positive note, both algorithms appear to be quite successful in creating two distinct groups of estimates by assigning much higher fitted coefficient values to the 10 variables chosen to influence the response, as opposed to the lower estimate values obtained for the other 90 design matrix columns. Ultimately, this will result in the original 10 variables having a much larger influence on the scores obtained for new observations, which is a welcome attribute. Finally, it can be seen that CPLEX produces

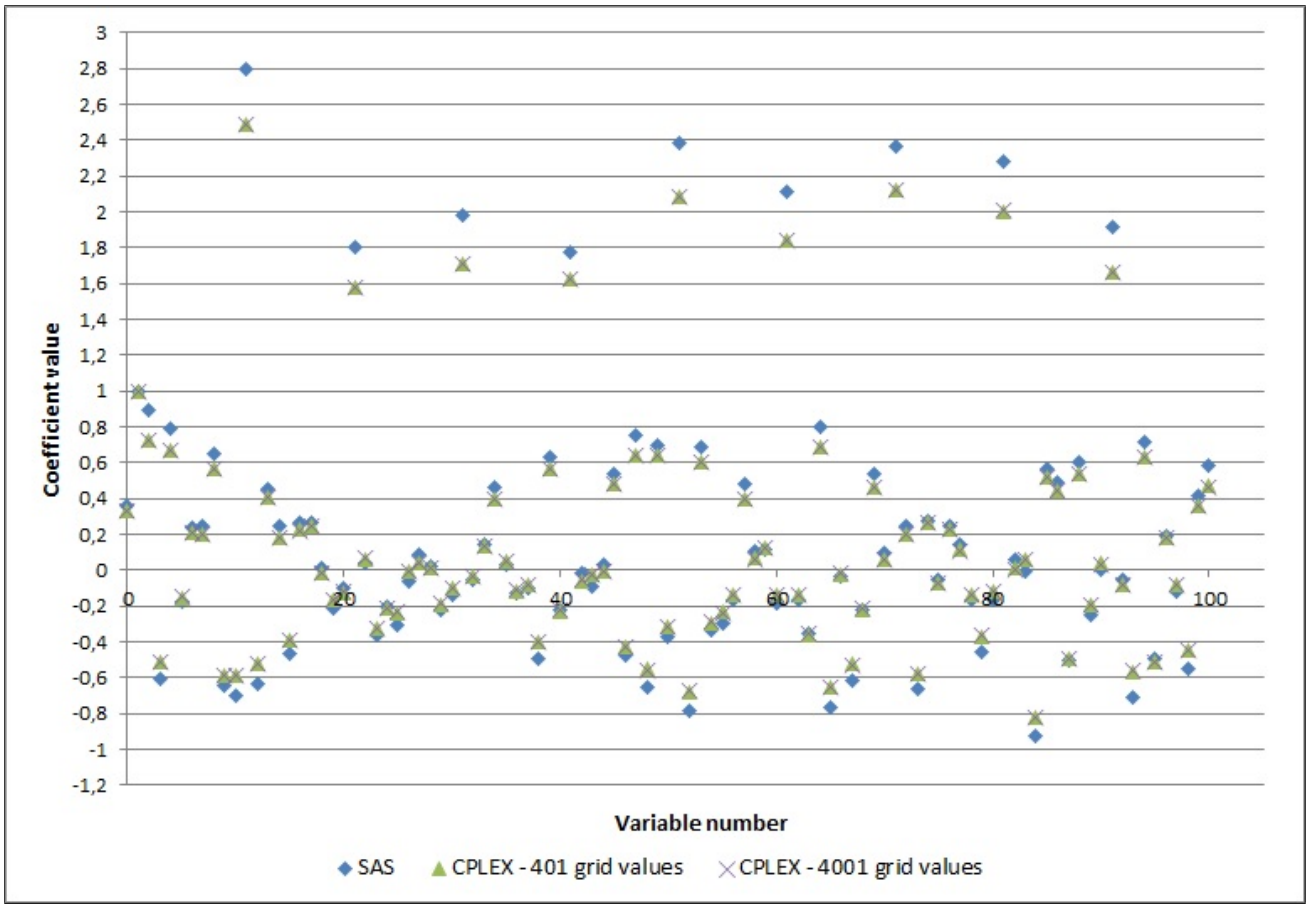


Figure 6.2: Regression parameter estimates obtained by SAS and CPLEX models when $\rho = 0.5$

estimated coefficients that are slightly closer to their true underlying, specified values as opposed to those fitted by SAS. The estimates of the 10 chosen variables are vaguely closer to one, while the fitted coefficients associated with the remaining 90 inputs are scattered marginally closer towards the zero-line. Figure 6.2 provides a graphical representation of estimated coefficients obtained by each model where $\rho = 0.5$.

In order to assess the validity of using (5.20)–(5.26) to facilitate variable selection in regression models, both SAS and CPLEX models are refitted to the dataset with $\rho = 0.5$, but a cardinality constraint q was added to the models during the second round of model execution. Since it is known that 10 equally spaced inputs were selected to influence the response, the cardinality parameter was set to $q = 10$, meaning that both models were instructed to have exactly 10 non-zero estimated parameters in their final solution vectors. Best subset selection was performed in SAS by using the SELECTION = SCORE option in the PROC LOGISTIC statement, along with the options START=10 and STOP=10. This allows SAS to only return the best 10-variable model. SAS utilises the *Leaps and Bounds* or LBA metod of Furnival and Wilson (1974) to

Table 6.2: Estimated regression coefficients in SAS and CPLEX when $q = 10$ and $\rho = 0.5$

Variable	CPLEX fitted coefficient	SAS fitted coefficient
X_1	0.9475	0.9579
X_{11}	1.1336	1.1360
X_{21}	0.8011	0.8157
X_{31}	1.0249	1.0249
X_{41}	1.0041	1.0119
X_{51}	1.0101	1.0253
X_{61}	0.9924	0.9927
X_{71}	1.3081	1.3076
X_{81}	1.0129	1.0081
X_{91}	0.9999	1.0035

perform best subset selection[2].

SAS attained a 10-variable model with a log-likelihood of -164.698, while the linearised model in CPLEX produced 10 estimates for the exact same variables chosen by the SAS best subset selection procedure, yielding a final likelihood value of -165.243. Both models selected non-zero estimates for the 10 initial variables chosen to have true regression coefficients of one and no significant difference was found between the two solution vectors. Furthermore, both models obtained final coefficient estimates that were very close to their true underlying values, with slight deviations for $X_{11}(\hat{\beta}_{11} = 1.1336)$, $X_{21}(\hat{\beta}_{21} = 0.8011)$ and $X_{71}(\hat{\beta}_{71} = 1.3081)$. Table 6.2 shows the estimated regression coefficients for the 10 selected variables in both models.

For the second and third correlation exercises, the correlation parameter ρ was specified to be 0.8 and 0.85, respectively, resulting in much more severe linear correlations being present in the training data. SAS displayed log-likelihood values of -78.789 ($\rho = 0.8$) and -89.862 ($\rho = 0.85$), whereas the linearised model concluded with objective function values of -82.181 for $\rho = 0.8$ and -94.473 for $\rho = 0.85$. As can be expected, the parameter estimates obtained by both models were noticably different from their true underlying values that were specified during the creation of the datasets, with the models failing to establish two distinct groups of estimates like those seen previously when the correlation coefficient was not as extreme. This resulted in a final solution vector where the fitted parameters seem to be far-removed from their true underlying values for both models. Once more, as in the case where $\rho = 0.5$, it was noted that the linearised model obtains fitted coefficients that are located marginally closer to their true specified values. Figure 6.3 plots the results for both models when $\rho = 0.8$.

[2]Refer to Appendix C for a detailed discussion on the LBA best subset selection approach.

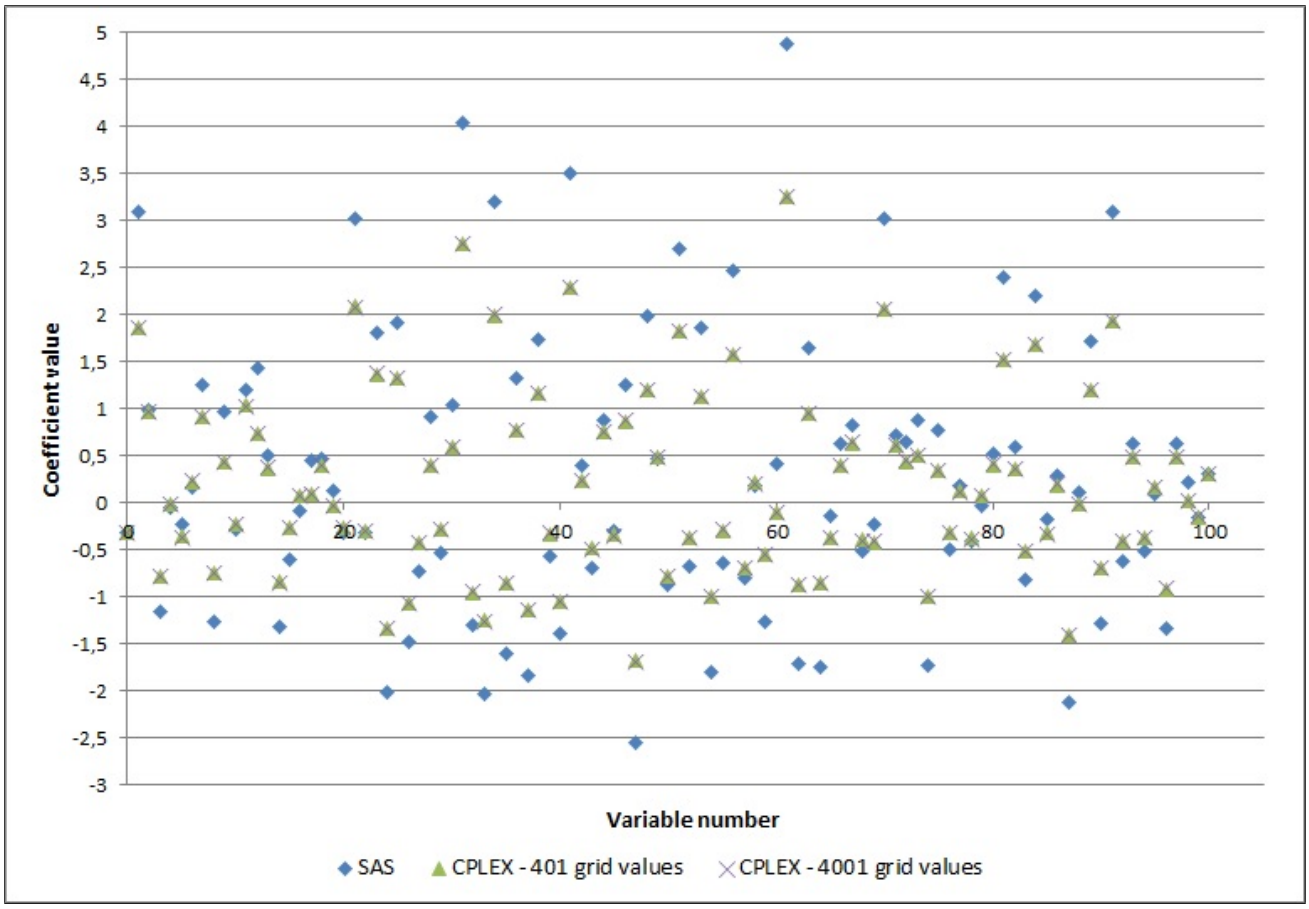


Figure 6.3: Regression parameter estimates obtained by SAS and CPLEX models when $\rho = 0.8$

6.3 Tests on real-world data: HEART dataset

In this exercise, the performance of the proposed linearised model is tested on a real-world dataset that is openly available for download. It should be noted that for this test, equation (5.16)–(5.19) is tested first, meaning that a linearised model without subset selection is fitted to the data. This is done to gauge whether or not the linearised model produces results comparable to the benchmark when applied to data that were not simulated. Afterwards, the proposed model in (5.20)–(5.26) is fitted in order to perform best subset selection on the same set of data.

Consider the HEART dataset that is readily available under the SASHELP library in SAS Enterprise Guide or Base SAS and contains the results of the Framingham Heart Study (SAS Institute Inc., 2017). The dataset comprises of 5 209 observations with various attributes. The response variable in question is called *Status* and is equal to 1 if the subject died and 0 if the subject survived. Of the 5 209 subjects, 3 218 survived ($Y = 0$) and 1991 died ($Y = 1$). Table 6.3 lists the 12 predictors that exist within the dataset (note that variables marked with an [*] were not supplied within the dataset, but were custom inputs that were created based on the

Table 6.3: List of predictor variables from the HEART dataset used as inputs in regression

Variable name	Description
AgeAtStart	Age of subject
Height	Height of the subject
Weight	Weight of the subject
Diastolic	Diastolic blood pressure reading
Systolic	The subject's systolic pressure reading
MRW	The subject's Metropolitan Relative Weight (similar to Body Mass Index)
Smoking	Number of cigarettes smoked by the subject (0 for non-smokers)
Cholesterol	The subject's cholesterol level
CHD_ind*	Indicates whether the subject was diagnosed with cardiovascular heart disease (1 if yes, 0 if no)
Smoke_ind*	Indicates whether the subject smokes or not (1 if yes, 0 if no)
BP*	Indicates whether the subject has high blood pressure or not (1 if yes, 0 if no)
Sex_bin	The gender of the subject (1 if male, 0 if female)

data that were available).

All variables that contained missing values (except for the *Smoking* variable) were imputed with the median value of the dataset for the predictor in question. In the case of the *Smoking* variable, the value was set equal to 0 if it was missing. Finally, using a 50/50 split, half of the dataset was selected at random to train the models. The other half was used to test the generalisation abilities of both models.

The solutions obtained from the linearised approach as well as SAS's PROC LOGISTIC procedure are nearly identical, with virtually no difference between the estimated regression coefficient vectors. In either case, the models choose relatively high coefficient estimates for the subject's age, whether the subject was diagnosed with cardiovascular disease, whether the subject smokes and the subject's gender, while setting the parameter estimates for the remaining variables close to zero. While the cardinality constraints discussed in equation (5.20)–(5.26) were not utilised during this round of model execution, the results suggest that if best subset selection was applied, the estimated coefficients of the remaining variables could have been set exactly equal to zero without much loss of accuracy. Figure 6.4 illustrates graphically the estimates produced by the linearised model in CPLEX (again, note that SAS's logistic procedure yielded identical solutions).

Since the solutions provided by both approaches are the same, it should come as no surprise that the models obtained from both SAS and CPLEX display similar misclassification rates and Gini coefficients. While the linearised model attains a 24.07% misclassification rate on the training set and 24.62% misclassification rate on the test set, the SAS logistic regression model displays misclassification rates of 24.14% and 24.39% on the training and test sets, respectively. A Gini[3] coefficient of 63.79 obtained from the CPLEX model ties with a 63.75 Gini value seen

[3]A Gini coefficient is often used in conjunction with or instead of the c-statistic or AUC (area under the curve), all of which are ranking tools utilised to measure model performance and have similar interpretations (Rezac and Rezac, 2011). Specifically, AUC = (Gini + 100)/2.

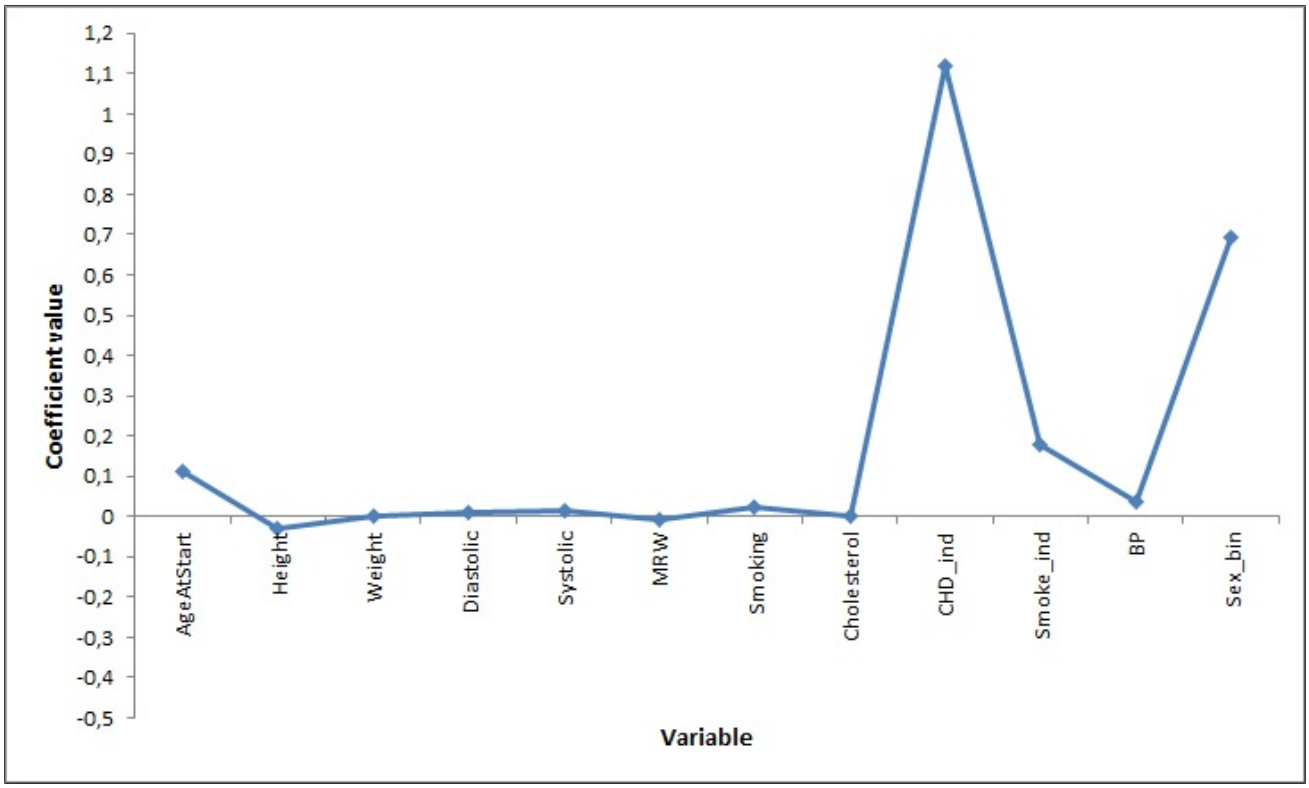


Figure 6.4: Coefficient estimates obtained by the linearised logistic regression model in CPLEX for the HEART dataset

for the SAS model when the ranking abilities of the models are evaluated on the training set, while Gini coefficients of 61.29 for the CPLEX model and 61.36 for the SAS model were achieved separately on the test set.

Next, the performance of the linearised model with best subset selection was tested. Since the approach outlined in (5.20)–(5.26) was being utilised during the second round of model fitting, it meant that the cardinality constraint q had to be explicitly specified. This was done by executing SAS's version of best subset selection by using the SELECTION = SCORE option in the PROC LOGISITIC model statement and selecting BEST = 1. By specifying these arguments, SAS will fit the best possible model containing l variables, for $l = 1 \dots p$, where a model with p non-zero estimated regression coefficients represents the full model with all of the input variables added to the final output. In other words, SAS finds the best 1-variable model, after which it finds the best 2-variable model and proceeds through the list of inputs by repeating this exercise until all of the design matrix columns are included in the last model iteration. After each model was fitted, the AUC was reported. This was done to determine the point at which the performance of the model starts to diminish, meaning that the addition of more variables to the mix does not improve model fit. Figure 6.5 illustrates that a model containing five input variables should suffice (indicated by the square marker). Notice that the AUC starts to level off beyond this

point, implying that all subsequent models would deliver similar predictive performance.

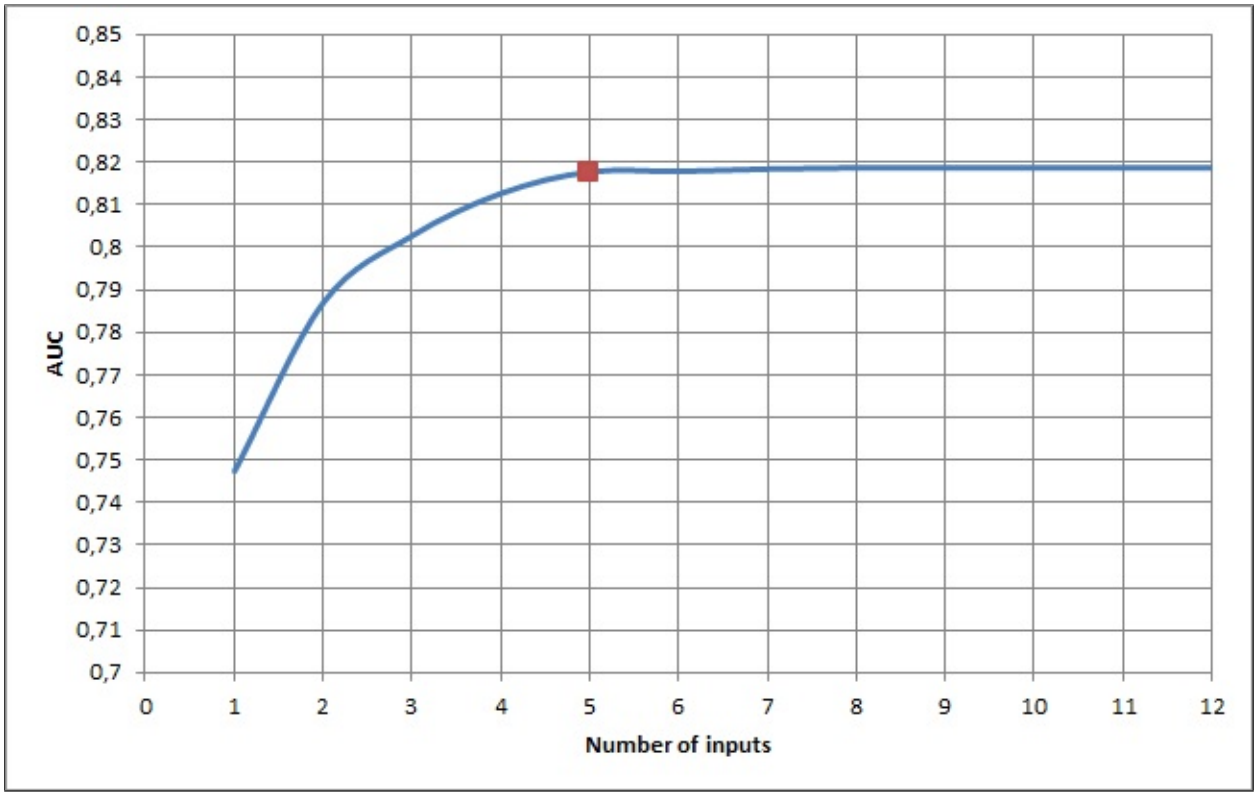


Figure 6.5: AUC for each model iteration applied to the HEART dataset

By specifying $q = 5 + 1$ (one more to make provision for the intercept term), the model in (5.20)–(5.26) was executed in CPLEX and yielded a maximum log-likelihood of -1305.83, while its counterpart fitted in SAS achieved a near-identical MLE of -1305.47. Both approaches opted to include the same five variables in the final model, along with estimated regression coefficients that are nearly indistinguishable for the selected inputs and the intercept term. Furthermore, accuracy statistics and Gini values were obtained which are strikingly similar to those seen previously when no feature selection was applied. This supports the assertion made earlier which states that excluding the remaining inputs in the design matrix (barring the five variables that were selected) will most likely not have an adverse affect on the model's predictive prowess.

Table 6.4 displays the estimated regression coefficients obtained by the two respective models when best subset selection is applied, while Table 6.5 contains model evaluation metrics. From Table 6.5 it can be seen that the linearised model with best subset selection outperformed the model fitted in SAS by a small margin.

Table 6.4: Coefficient estimates obtained for the HEART dataset during best subset selection

Variable	CPLEX fitted coefficient	SAS fitted coefficient
Intercept	-8.8826	-8.8905
AgeAtStart	0.1096	0.1096
Systolic	0.0185	0.01849
Smoking	0.0301	0.03016
CHD_ind	1.1009	1.1023
Sex_bin	0.6170	0.6175

Table 6.5: Model evaluation metrics for the HEART dataset based on models with best subset selection applied

Evaluation metric	CPLEX	SAS
Train Gini	63.79	63.74
Test Gini	61.36	61.33
Train Accuracy	75.93%	75.85%
Test Accuracy	75.77%	75.61%

6.4 Tests on real-world data: JUNKMAIL dataset

For the second real-world exercise, a dataset within the SASHELP library called JUNKMAIL was used. The data were collected by Hewlett-Packard labs (HP Computers) and subsequently donated by George Forman. The set holds data relating to 4 601 emails that were classified as SPAM ($Y = 1$) or PERSONAL / WORK-RELATED ($Y = 0$). The data are also openly available on the UCI Machine Learning Repository. There are 57 numeric predictor variables, most of which relate to metrics that indicate how frequently a certain word occurs within the email itself. Additionally, the last three variables are considered to be run-length attributes and measure the length and sequences in consecutive capital letters contained within the email. Already provided in the data was an indicator column dubbed *Test*, which splits the data into a model training set and a test set. The training set contains 3 065 observations, of which 1 847 correpsond to $Y = 0$, while 1 218 have a response of $Y = 1$. This particular dataset was chosen due to the fact that it contains a moderate number of observations (4 601) and a relatively large number of columns (57 predictors), which makes it more relatable and representative of an actual dataset that a user might encounter in daily life.

Exploratory data analysis revealed that most of the numeric variables contained a large number of zeros as entries, with the majority of inputs consisting only 30% or less of non-zero values (with some containing zero-values for as much as 96% of the observations). For this reason, the dataset was altered by changing each input into a binary variable. Every binary variable was given a value of 0 if the entry of the original input was 0 and a value of 1 if its entry was greater than 0. The binary variables were named by adding the prefix *bin* in front of the original

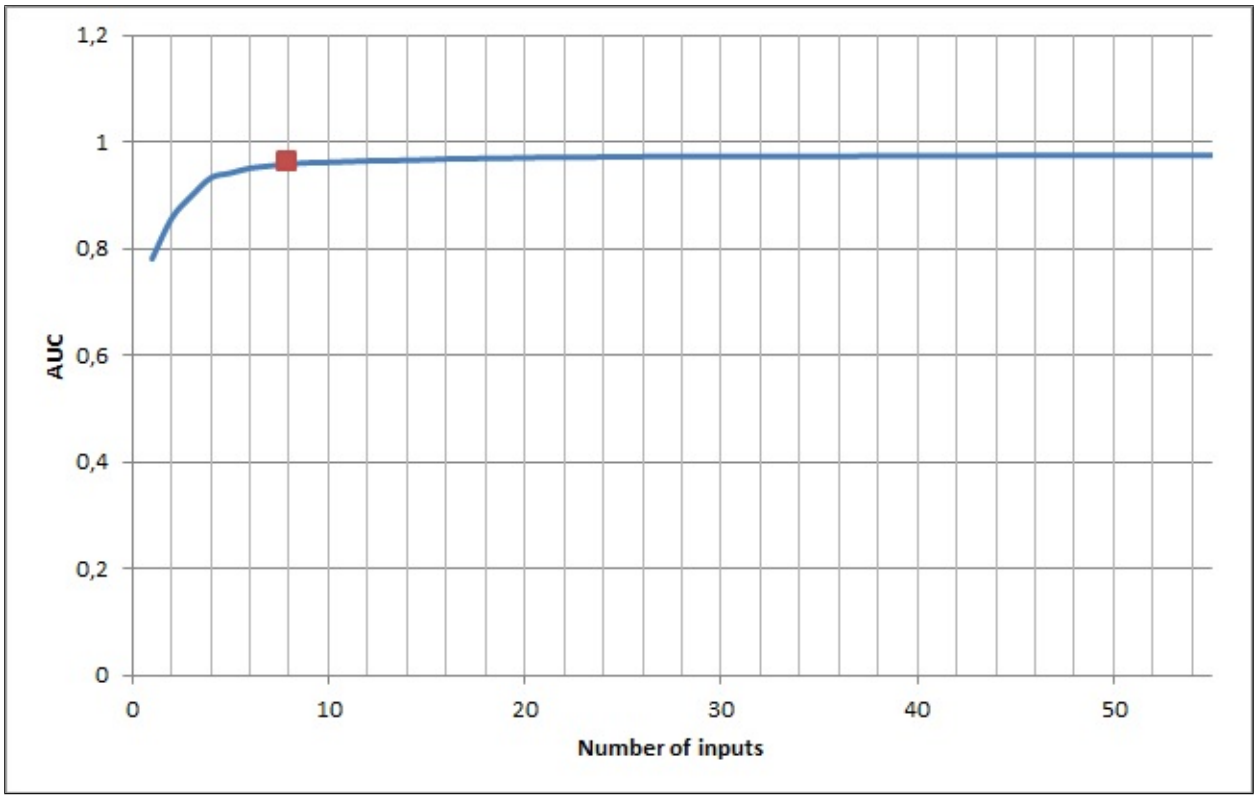


Figure 6.6: AUC for each model iteration applied to the JUNKMAIL dataset with binary predictors

variable's name, i.e. if the original input was called $DOLLAR$, the binary version would be called bin_DOLLAR. Both models in SAS and CPLEX were then fitted using the binary predictors as inputs instead of the original variables. The only columns that remained unchanged (no binary indicators were created for them) were $CAPAG$, $CAPLONG$ and $CAPTOTAL$. These variables related to the running length of capital letters in the email and did not contain any zeros. Ultimately, each of the binary variables can therefore be seen as an indication of whether or not a certain word occurred within the mail.

To determine an appropriate value for the cardinality parameter q, the same approach that was followed in Section 6.3 when best subset selection was carried out for the HEART dataset, was applied here. Once more, the best l-variable model, for $l = 1, \ldots, p$, was iteratively produced in SAS using PROC LOGISTIC with the SELECTION = SCORE option, after which the AUC for each model fit was recorded. A suitable candidate for the cardinality constraint in best subset selection was determined by using the value of q that corresponds to the point at which no real improvement in AUC can be seen. Figure 6.6 shows that a model with roughly eight input variables should be adequate (refer to the square marker).

The linearised model with $q = 8$ plus one more to account for the intercept, employed a grid spanning [-30, 30] with 0.01 intervals and its optimality gap was set to 5% in order to aid

Table 6.6: Coefficient estimates obtained for the JUNKMAIL dataset

SAS		CPLEX	
Variable	Coefficient	Variable	Coefficient
Intercept	-2.3365	Intercept	-2.2
bin_DOLLAR	2.173	*bin_DOLLAR*	2.3
bin_EDU	-3.1087	*bin_EDU*	-3.1
bin_EXCLAMATION	1.54	*bin_EXCLAMATION*	1.6
bin_FREE	1.5861	*bin_FREE*	1.6
bin_HP	-3.7084	*bin_HP*	-4
bin_MEETING	-2.976	*bin_MEETING*	-3.2
bin_REMOVE	2.757	*bin_REMOVE*	2.7
bin_YOUR	0.9179	*bin_OUR*	1

execution times. Both the linearised model and model in SAS produce solution vectors that are alike for the majority of the variables, noting that seven out of the eight non-zero fitted coefficients correspond to the same inputs. It can also be seen that the estimated regression coefficients obtained for these seven inputs are very close to one another within the two solution vectors, with minor differences existing between the two sets. The only exception is where SAS fits a non-zero regression estimate to the variable *bin_YOUR*, the linearised model in CPLEX chooses a non-zero coefficient for *bin_OUR*. While these are, in fact, different variables, both obtained estimated regression parameters of roughly one (with $\hat{\beta}_{bin\_OUR}$ being exactly equal to one). Table 6.6 displays the coefficients obtained by the two models.

When considering the accuracy of the models produced by the two respective techniques, it was found that the linearised model obtained a higher objective value of -740.66 for the log-likelihood as opposed to the slightly lower -742.449 produced by its counterpart in SAS. The final model fitted by SAS had a fractionally higher Gini statistic of 92.13 for the training set, as opposed to a Gini of 92.09 associated with the linearised model with best subset selection. Alternatively, the CPLEX logistic regression model obtained marginally better results for the test dataset, with a Gini of 91.37 being higher than the 90.91 seen when SAS's regression model is applied to the same test set. In terms of misclassification rates, both models obtained an accuracy statistic of 91.06% for the training dataset. However, once again, the linearised model appeared to fair better than the model produced by SAS when test set accuracy was considered. While the model fitted in CPLEX showed a misclassification rate of 8.59% for the test dataset, SAS's final model obtained a misclassification rate of 9.38% for the same set of data. The very small discrepancies that exist between the Gini statistics and accuracy measures of the training and validation sets prove that both models appear to be quite stable.

In an attempt to understand why the two separate models are not producing identical solution vectors, due to their inclination to have at least one predictor that differs in the subset of variables included in the final model, both models were refitted again. During each instance of

model fitting, a different value was specified for the cardinality parameter q. The initial value of q was set to be relatively low and subsequently increased in an ordinal fashion after each run.

It was noted that up until $q = 6$, both models yield identical solutions. At $q = 6$, the two models choose to include bin_DOLLAR , bin_EDU, $bin\_EXCLAMATION$, bin_FREE, bin_HP and bin_REMOVE. While small differences exist between the fitted regression parameter vectors produced by the respective models, the estimated coefficients for these six inputs were largely the same when comparisons were made. It should come as no surprise that both models achieved an objective function value for the log-likelihood as well as Gini values and misclassification rates for the training and test datasets that were nearly identical. At $q = 7$, SAS's logistic procedure opts to fit a non-zero coefficient to the variable bin_YOUR, whereas the linearised model in CPLEX chooses to include the variable $bin\_MEETING$ instead. When $q = 8$, the SAS logistic procedure concedes to choose $bin\_MEETING$ for inclusion in the final model (which was selected by the CPLEX model at $q = 7$ in the previous step). However, the CPLEX model still chooses not to fit a non-zero coefficient to the variable bin_YOUR (which was chosen by SAS at $q = 7$) and instead opts for the inclusion of the input bin_OUR (as was seen earlier on in this section). When the cardinality parameter was set to $q = 9$, it was found that both models elected the variable bin_1999 for inclusion at this step. Again, as with $q = 8$, the solution vector produced by SAS contained bin_YOUR in the final model, whereas the CPLEX solution vector contained bin_OUR instead. Finally, at $q = 10$, the logistic regression model fitted in SAS selected the variable bin_OUR as an addition to the final model – an input that was selected by the CPLEX model much earlier on. However, the model produced by CPLEX still chooses to neglect the variable bin_YOUR, which was already selected by SAS's logisitic procedure at $q = 7$, and instead chooses the predictor bin_RE to be part of the final solution.

Table 6.7 shows the log-likelihood values obtained along with model accuracy measures calculated for the training and test sets, respectively, for different values of the cardinality parameter q.

Table 6.7: Model accuracy statistics for training and test sets for different values of q

	SAS					CPLEX				
q	$\log L$	Train Gini	Test Gini	Train Accuracy	Test Accuracy	$\log L$	Train Gini	Test Gini	Train Accuracy	Test Accuracy
$q = 6$	-796.74	90.59	90.08	90.11%	89.97%	-796.85	90.60	90.06	90.11%	89.97%
$q = 7$	-774.50	91.35	91.67	90.34%	90.17%	-763.08	91.47	91.05	90.34%	90.17%
$q = 8$	-742.45	92.13	90.91	91.06%	90.63%	-740.66	92.09	91.37	91.06%	91.41%
$q = 9$	-723.87	92.62	91.65	91.58%	90.95%	-720.95	92.49	92.02	91.42%	91.80%
$q = 10$	-708.72	92.67	91.67	91.32%	91.67%	-705.35	92.93	92.89	91.81%	91.60%

Table 6.8 displays the regression parameter estimates produced by the two models for different cardinality constraints.

Table 6.7 reveals that the logistic regression models fitted by using the linearised model in CPLEX consistently produced log-likelihood values along with Gini and accuracy statistics that were better than those obtained by in SAS in most cases, regardless of the value of q.

Table 6.8: Estimated regression coefficients obtained for different values of q

	SAS					CPLEX				
Variable	$q = 6$	$q = 7$	$q = 8$	$q = 9$	$q = 10$	$q = 6$	$q = 7$	$q = 8$	$q = 9$	$q = 10$
Intercept	-2.1601	-2.4335	-2.3365	-2.2671	-2.335	-2.2	-2.1	-2.2	-2.15	-2.0333
bin_DOLLAR	2.3756	2.1352	2.173	2.2508	2.1846	2.4	2.4	2.3	2.35	2.5
bin_EDU	-3.3352	-3.4033	-3.1087	-2.989	-2.9616	-3.3	-3	-3.1	-2.9	-2.7
bin_EXCLAMATION	1.689	1.5476	1.54	1.5399	1.4633	1.7	1.7	1.6	1.55	1.6333
bin_FREE	1.6817	1.4807	1.5861	1.6956	1.5818	1.7	1.8	1.6	1.75	1.8
bin_HP	-3.7405	-3.7459	-3.7084	-3.1803	-3.3536	-3.7	-3.7	-4	-3.4	-3.46667
bin_REMOVE	3.0102	2.8466	2.757	2.7048	2.5401	3	2.9	2.7	2.65	2.6333
bin_YOUR		0.9324	0.9179	0.9173	0.7353					
bin_MEETING			-2.976	-2.8226	-2.9963		-3	-3.2	-3.05	-3.06667
bin_OUR					0.8855			1	1.05	1.1
*bin_*1999				-1.4727	-1.5279				-1.55	-1.5333
bin_RE										-0.96667

Lastly, empirical results in Sato et al. (2015) show that the linearised approximation with variable selection proposed by the authors selected between 28 and 31 variables for inclusion in the final model when the approach in (5.12)–(5.15) was applied to the JUNKMAIL dataset. Unfortunately, the authors do not report on model evaluation metrics such as the misclassification rate, Gini statistic or AUC. Additionally, instead of splitting the modelling dataset so as to perform training versus validation exercises, Sato et al. (2015) chose to utilise the full JUNKMAIL dataset during model fitting. As a result, a comprehensive comparison of the different linearised approaches discussed in Chapter 5 could not be carried out. However, it can be seen that the model in (5.12)–(5.15) selected more than three times the number of variables when compared to the linearised model proposed in (5.20)–(5.26). Furthermore, it is also observed that the eight-variable model shown in this section displays impeccable levels of accuracy while simultaneously maintaining a high level of descriptiveness by parsimoniously including a lower number of predictors. In spite of the favourable results achieved in this section, it would be naive to assume that the differences that exist between the model fitted in Sato et al. (2015) and the model shown on the previous page can entirely be attributed to the use of (5.20)–(5.26) instead of (5.12)–(5.15). Instead, it should be aknowledged that an indispensable amount of value is added to the quality of the models by exploratory data analysis and appropriate variable transformations.

6.5 Test on high dimensional data

For this exercise, best subset selection was performed on a dataset where the number of input variables is substantially large. By using the same approach described in Section 6.1, a dataset containing $n = 1\ 000$ cases and $p = 500$ standard normal random variables was generated. Of the 500 effects, a total of 15 equally spaced predictors were chosen to influence the response and have true regression coefficients equal to one, meaning that $\beta_l^* = 1$ where $l \in Q$ and

$Q = \{1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 131, 141\}$. The remaining 485 inputs were specified to have $\beta_j^* = 0$ with $j \notin Q$ and had no affect on the repsonse.

As stated in previous chapters, best subset selection becomes extremely resource intensive when $p \gg 40$ due to the exponential increase in execution time for each additional variable considered by the model. For this reason, both models in SAS and CPLEX were only allowed to execute model fitting for a predetermined amount of time. Once again, the model in (5.20)–(5.26) was used to facilitate best subset selection in CPLEX, while the SELECTION = SCORE with START = 15 and STOP= 15 options were specified in SAS PROC LOGISTIC.

After only two hours of execution, the linearised model in CPLEX already selects the correct 15 variables for inclusion by setting $z_l = 1$ where $l \in Q$. Furthermore, all other variables are omitted from the final model due to $z_j = 0$, $j \notin Q$, in the CPLEX solution. Terminating the CPLEX logistic regression model after two hours yields a log-likelihood value of -276.762 with an optimality gap of 78.70%[4]. The exact same model is obtained when CPLEX is allowed to terminate after four hours instead of two, with an identical objective function value of -276.762. However, the optimality gap had reduced to 71.35%. In an attempt to find a model that is closer to optimality, the linearised model in CPLEX was applied once more to the data with a specified run time of 48 hours. As with the models produced after two and four hours, respectively, coefficient estimates greater than zero are obtained for the correct 15 variables, while all other parameter estimates are set equal to exactly zero. Once again, a maximum likelihood estimate of -276.762 is produced. In spite of identical log-likelihood values, a significantly lower optimality gap of 58.68% is observed. Table 6.9 shows the parameter estimates obtained by the linearised logistic regression model in CPLEX. Investigation revealed that all three of the aforementioned models yielded the same estimated coefficient values regardless of the execution times that were specifed. Notice that the regression coefficients which were estimated for the 15 variables are sufficiently close to their true underlying values of one.

In contrast to the linearised model in CPLEX, SAS fails to yield any results after 48 hours of execution time. However, this can be expected, as best subset selection in SAS is considered to be computationally impractical when $p > 75$ (Lund, 2017). Several course notes on regression modelling in SAS, such as Patetta (2012) and Patetta et al. (2009), suggest that modellers should opt for stepwise variable selection procedures in such cases instead.

The results indicate that the linearised model yields sufficient output and can adequately perform best subset selection when the number of predictors p is large. Specifically, the model in (5.20)–(5.26) tends to select the appropriate inputs and produces the correct underlying model even when execution is stopped early. Empirical studies show that (5.20)–(5.26) finds the correct model and parameter estimates early on, but requires significantly more time to prove optimality. This is supported by the fact that all CPLEX runs obtained identical log-likelihood objective

[4]Refer to Section 4.2.2 of Chapter 4 for the definition of an optimality gap.

Table 6.9: Parameter estimates produced by the linearised model in CPLEX for $p = 500$ and $q = 15$

Coefficient	Estimate
β_0	0.0137
β_1	1.2034
β_{11}	1.1285
β_{21}	0.9799
β_{31}	1.1253
β_{41}	0.9263
β_{51}	0.8725
β_{61}	1.1161
β_{71}	0.9909
β_{81}	1.1440
β_{91}	1.0676
β_{101}	1.1994
β_{111}	1.1556
β_{121}	1.1341
β_{131}	0.9378
β_{141}	0.9505

function values and estimated coefficients, but exhibited a declining optimality gap alongside an increase in run time. Bertsimas et al. (2016) encountered the same phenomenon, noting that their suggested MIO formulation for best subset selection in linear regression produced suitable models fairly quickly, but needed notably more time to provide a guarantee on optimality.

6.6 Choice of grid range and number of grid values

When the linearised model was fitted to both simulated and real-world datasets, it became apparent that the range specified for the grid used in the linearisation is of a subjective nature and problem-specific. Indeed, much wider ranges needed to be used for the JUNKMAIL dataset where the estimated coefficients could potentially take on values much larger than one, as opposed to simulated data where the inputs were standardised. However, it was found that if the true underlying regression parameters are close to one or relatively small, the following range for the grid can be employed:

$$[-(1+b)q, (1+b)q], \tag{6.1}$$

where q is the number of variables that have an influence on the response with the true underlying regression coefficients being non-zero. If q is unknown, it can be replaced with p. The parameter b is a buffer used to allow the model to deviate somewhat from the values of the true underlying coefficient parameters when fitting the model, i.e. to allow for white noise. During the exercises conducted on simulated data (as discussed in Sections 6.1 and 6.2), $b \leq 1$ was found to be more

than adequate.

Alternatively, if no indication exists as to what values the true underlying regression parameters may take on (as is the case with most real-world applications), a possible starting point would be to fit the full logistic regression model without variable selection (the choice of model does not matter, so either the linearised model in (5.16)–(5.19) or nonlinear model in (2.11) with Newton-Raphson method can be employed). Once the parameter estimates are obtained, calculate $M_1 = \min\{\sum_{l=1}^{p+1} \hat{\beta}_l X_{1l}, \ldots, \sum_{l=1}^{p+1} \hat{\beta}_l X_{nl}\}$ and $M_2 = \max\{\sum_{l=1}^{p+1} \hat{\beta}_l X_{1l}, \ldots, \sum_{l=1}^{p+1} \hat{\beta}_l X_{nl}\}$, where n is the number of observations in the modelling set. The following incarnation of (6.1) can then be used as potential range for the grid:

$$[M_1 - b_1, M_2 + b_2], \tag{6.2}$$

where b_1 and b_2 have the same interpretation as b in (6.1) and are not necessarily equal to one another (however, they may be). Note that when (6.2) is used, slightly larger values for b_1 and b_2, with $b_1 > 1$ and $b_2 > 1$, need to be specified. For the real-world datasets presented in this chapter, $5 \leq b_{opt} \leq 20$ with $b_{opt} = b_1 = b_2$ was found to be a suitable choice. Occasionally, a situation may present itself where M_1 or M_2 (or both) is unreasonably large. This is often caused by a select few outliers that are present in the data. If M_1 or M_2 is significantly inflated, it may lead to extended model execution times (if more grid values are used) or a linearised logistic regression model that severely underestimates the nonlinear log-likelihood (if k is not increased to compensate for the wider grid). To combat this, the user may opt to utilse $M_1^{\alpha} = P_{\alpha}$ and $M_2^{\alpha} = P_{(1-\alpha)}$ instead, where P_{α} denotes the $(\alpha \times 100)$-th percentile of the set $\{\sum_{l=1}^{p+1} \hat{\beta}_l X_{1l}, \ldots, \sum_{l=1}^{p+1} \hat{\beta}_l X_{nl}\}$. While $\alpha = 0.05$ serves as a common middle ground, more conservative modellers might consider using $\alpha = 0.01$, whereas $\alpha = 0.1$ may be employed by users who are less concerned with model accuracy and more focussed on speed of execution.

The number of grid values, k, used in (5.20)–(5.26) remains a more subjective issue. As mentioned previously, as k tends to infinity, the linearised model resembles the nonlinear log-likelihood function. Therefore, increasing k should result in improved estimates and, subsequently, a more accurate model. However, experimental results have revealed that a substantial increase in k results in a rapid increase in model execution time, but does not necessarily deliver a significant improvement in model fit. Since the constraint matrix contains $n(k+2)$ rows for (5.16)–(5.19) and $n(k+2)+2(p+1)+1$ rows for (5.20)–(5.26), it is clear that an environment where both n and k are substantial would result in a much larger number of constraints that need to be considered by the solver, which has an adverse effect on model run time. Since n and p are fixed, the only other option would therefore be to reduce k.

Chapter 7, presented next, examines the trade-off that exists between the quality of the solutions produced by the MILP formulation and model execution time for different values of k.

Chapter 7

Improvements in computing time

7.1 Introduction

As stated before, best subset selection is an NP-hard problem, which is often considered to be resource intensive and is regularly abandoned in favour of more computationally friendly approaches, such as stepwise regression. However, the results in this chapter will demonstrate that the application of the MILP formulation in (5.20)–(5.26) with the appropriate settings allows the modeller to obtain solutions that are closer to optimality within an increasingly shorter time frame, even in the case of large datasets. In fact, improvements in model run time and/or optimality can be achieved by simply specifying a suitable value for the number of grid values, k, utilised by the linearised approximation of the log-likelihood function and by considering the optimality gap of the MILP prior to execution.

By specifying that the optimality gap must be greater than zero, significantly improved execution times can be achieved. In Chapter 4, the optimality gap was introduced and discussed in detail. Recall that the gap refers to the difference between the best lower bound solution and the best upper bound solution, which is subsequently expressed as a percentage. The default optimality gap in the mathematical solver CPLEX is usually set equal to a very small number that is close to zero, such as $1e-5$. A gap of zero will mean that the solution yielded by the model can be considered as the most optimal possible solution attainable and cannot be improved. However, a smaller optimality gap that is closer to zero might result in a situation where the solver takes longer to terminate, which in turn can lead to prolonged model execution times. Ultimately, extended model run times in the presence of a small gap can be attributed to the fact that the solver has to evaluate more nodes during its execution in order to prove optimality. Empirical evidence in Chapter 6, as well as results presented by Bertsimas et al. (2016), suggest that the MILP regression model finds the correct solution fairly quickly during the early stages of model fitting (when the gap is still relatively large), but requires additional time in order to prove optimality. Therefore, the user can comfortably request the model to terminate

at a larger optimality gap, knowing that the resulting solution will be close to optimality (or may even be the optimal solution). All of the results presented in this Chapter were obtained from models that were executed where the gap was set equal to 5% (unless stated otherwise), meaning that the solutions found by these models are at most 5% worse than the most optimal solution.

Finally, recent advances in computing power, which has enabled solvers to take on much larger and more complex problems, should also be acknowledged. Refer to Section 4.2.3 in Chapter 4, where improvements in arithmetical power and mathematical programming algorithms are briefly addressed.

In this chapter, experiments were performed by fitting the linearised model in (5.20)–(5.26) to the data for different values of k in order to investigate the trade-off between model run time and quality of the results produced. Specifically, logistic regression models were iteratively executed on the same set of data for $k \in \{401, 161, 81, 41\}$. It should be noted that the maximum number of grid values employed by all models in this section was set equal to 401 and that the solutions obtained when $k = 401$ were used as a baseline reference. This means that the execution times of and results yielded by all other models that were fitted to the same dataset for $k \neq 401$ were compared to those found for the baseline model. This is motivated by the results obtained in Chapter 6, where it was found that the linearised logistic regression model with $k = 401$ produced maximum likelihood estimates that were sufficiently close to the MLE obtained by the nonlinear model in SAS. Additionally, it was observed that using much larger values for k unnecessarily increased model execution times without showing significant improvement in model fit.

7.2 Tests on simulated data

Consider once more the simulated datasets that were used in the modelling exercises conducted in Section 6.1 of Chapter 6 and recall that various training sets containing n observations, with $n \in \{1\ 000, 3\ 000, 5\ 000, 10\ 000\}$, and 100 variables from a standard normal distribution were generated. In all of these datasets, 10 equally spaced features were chosen to influence a latent, continuous response variable $\mathbf{Y}^c$ and were specified to have a true regression coefficient of one, or $\beta_l^* = 1$ for $l \in Q$, where $Q = \{1, 11, 21, 31, 41, 51, 61, 71, 81, 91\}$. A binary outcome vector was subsequently created from $\mathbf{Y}^c$ and served as the target variable for the logistic regression model. By knowing that the final model should contain 10 predictors, the model in (5.20)–(5.26) was fitted to the data whereby q was set equal to 10 plus one more to compensate for the intercept. Table 7.1 displays the maximum likelihood estimates obtained during each model fit for different values of k, along with the corresponding time consumed by every round of model execution.

Table 7.1 shows that a significant decrease in model execution time can be achieved by

Table 7.1: Linearised logistic regression model performance for different values of k

	$\log L$				Time (in seconds)			
k	$n = 1\ 000$	$n = 3\ 000$	$n = 5\ 000$	$n = 10\ 000$	$n = 1\ 000$	$n = 3\ 000$	$n = 5\ 000$	$n = 10\ 000$
401	-321.037	-1010.981	-1755.168	-3500.658	1727.93	3231.42	25148.31	102583.00
161	-321.459	-1012.392	-1757.545	-3505.505	394.76	1803.31	5060.53	9531.11
81	-323.021	-1017.388	-1766.271	-3522.718	217.31	810.53	1964.24	6657.11
41	-329.252	-1037.018	-1800.191	-3589.838	99.75	322.35	1105.39	5467.77

reducing k. However, shrinking the number of grid values employed by the linearised model does not appear to have an adverse effect on the MLE obtained by the model. Table 7.2 shows by how much the optimum solution has weakened (in percentage terms) alongside smaller k values. It can be observed that specifying lower values for k seem to exert very little influence on the quality of the solution obtained.

Table 7.2: Deterioration in $\log L$ for different values of k

k	$n = 1\ 000$	$n = 3\ 000$	$n = 5\ 000$	$n = 10\ 000$
401	-	-	-	-
161	-0.13%	-0.13%	-0,13%	-0,13%
81	-0.61%	-0,63%	-0.63%	-0.63%
41	-2.55%	-2.57%	-2.56%	-2.54%

From the results displayed in Tables 7.1 and 7.2 it can be seen that the amount of time required to perform logistic regression modelling with best subset selection may be reduced by up to 90% if the modeller is willing to accept a final solution that is less than 3% weaker. Lastly, it should be noted that all of the models selected the correct 10 variables for inclusion in the final solution, regardless of the value specified for k. Furthermore, parameter estimates obtained for these 10 inputs were sufficiently close to their true underlying values of one, i.e. $\hat{\beta}_l \approx 1$ for $l \in Q$, after each round of model fitting.

7.3 Tests on real-world data

Modelling exercises similar to those presented in the previous section were conducted on real-world datasets as well. Note that model performance for the HEART dataset in Section 6.3 of Chapter 6 was not considered in this section, as the modelling set contains only 12 potential predictor variables, which can easily be accommodated by most variable selection techniques. Instead, two additional real-world datasets were studied alongside the JUNKMAIL dataset, namely the ONLINE NEWS POPULARITY dataset of Fernandes et al. (2015) and the SUPERCONDUCTIVITY DATA dataset of Hamidieh (2018). Both datasets, along with the JUNKMAIL dataset, were chosen for inclusion in this section due to their relative size. The number of

rows, n, as well as the dimensionality, p, of these datasets are sufficiently large and are more representative of an actual set of data that a modeller might encounter in practice. All of the aforementioned data are available for download from the UCI Machine Learning Repository.

The ONLINE NEWS POPULARITY dataset contains a total of $n = 39\ 640$ records and 61 columns, of which 58 are potential predictors. Of these 58 columns, 4 were found to be linear combinations of other features in the dataset and were subsequently removed, leaving 54 variables that were eligible for inclusion in the final model. The target variable in question is called *Shares* and captures the number of online shares of a news article upon publication. It was noted that the response vector is a numeric field which contains integer entries, where *Shares* $\in \mathbb{Z}^+$, which implies that it cannot be used as-is in binary classification problems such as logistic regression. Instead, a suggestion given by Fernandes et al. (2015) was applied whereby a news article was classified as popular when it had more than 1 400 shares. Alternatively, an article with a lower number of shares was seen as unpopular. This lead to the creation of the binary outcome variable $\mathbf{Y} \in \{0, 1\}$ for logistic regression by specifying

$$Y_i = \begin{cases} 0 & \text{if } \textit{Shares} \ < 1400, \\ 1 & \text{if } \textit{Shares} \ \geq 1400, \end{cases}$$

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

The final dataset contained a total of 18 490 records with $Y = 0$ and 21 150 cases with $Y = 1$ after the target variable was transformed. As is the case within most predictive modelling settings, the entire dataset containing almost 40 000 observations was not used for model training. A smaller training set consisting of $n_1 = 10\ 000$ cases was created by randomly sampling 5 000 observations with $Y = 1$ and 5 000 observations with $Y = 0$ from the total set. The remaining $n_2 = 29\ 640$ were used as a validation set to test the generalisation abilities of the model. Lastly, studying the AUC curve produced by models fitted to the dataset revealed that the model becomes saturated when approximately 22 predictors are included in the final solution. Therefore, the model in (5.20)–(5.26) with best subset selection was executed on the training set by using $q = 22 + 1$ as a cardinality parameter (one more to compensate for the intercept term in regression)[1].

The SUPERCONDUCTIVITY DATA dataset contains information relating to the chemical compound of superconductors, along with the critical temperature achieved by the superconductor which is captured in a column named *critical_temp* and serves as the response variable of interest. The table comprises of $n = 21\ 263$ observations and 81 features that can be used as potential predictors in the final model. As is the case with the ONLINE NEWS POPULAR-

[1]Recall that the same methodology was used to select a suitable value for q when logistic regression modelling with best subset selection was carried out for the HEART and JUNKMAIL datasets in Chapter 6.

ITY dataset, the outcome variable had to be discretised in order to formulate a suitable target variable for logistic regression, since the field *critical_temp* is a positive, real-valued number. In a seperate study wherein machine learning models were trained to predict the critical temperature of superconductors, Stanev et al. (2018) notes that superconductors can be adequately divided into two groups of materials associated with either high or low temperatures by using *critical_temp* = 10K as a cut-off. A binary response variable was subsequetly created as follows:

$$Y_i = \begin{cases} 0 & \text{if } \textit{critical\_temp} \ < 10, \\ 1 & \text{if } \textit{critical\_temp} \ \geq 10, \end{cases}$$

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

Once the transformation shown above was carried out, the final dataset encompassed a total of 7 729 observations with $Y = 0$ and 13 534 records with $Y = 1$. Once more, a smaller dataset was created for the purposes of training the model. Results presented by Stanev et al. (2018) show that satisfactory levels of accuracy can be achieved when predicting superconductor temperatures with a modelling set that contains roughly 3 000 to 6 000 observations. For this reason, a training dataset with $n_1 = 6\ 000$ observations was composed by randomly sampling 3 000 cases with $Y = 0$ and 3 000 records with $Y = 1$. The other $n_2 = 15\ 263$ observations that were excluded from the training set were kept as a test set.

In order to decide on a suitable value for the cardinality parameter q, consideration was given to the results presented by Hamidieh (2018) and Stanev et al. (2018) instead of having to draw up and analyse the AUC curve for 81 variables. Hamidieh (2018) opts to fit an extreme gradient boost model to the data, where the untransformed response variable *critical_temp* is used as a target. The author then proceeds to report the objective function gains achieved for each input variable as a percentage of the total objective function improvement obtained when all features are included in the model. By studying the individual gain of the top 20 predictors in the model, it was found that the improvement in the final objective function value seems to diminish between the first 10 to 15 variables. In turn, the emprical evidence supplied by Stanev et al. (2018) indicate that the performance of their classifiers achieve a plateau for models consisting of approximately 10 or more predictors. After analysing the results presented by both of the aforementioned authors, a value of $q = 10 + 1$ was used when performing best subset selection on the superconductors dataset in order to obtain a final model with 10 inputs and an intercept.

Recall that the exercises conducted in Section 6.4 of Chapter 6 involved the execution of the linearised logistic regression model with best subset selection for the JUNKMAIL dataset where a variety of values were specified for the cardinality parameter q, which produced solutions containing between $q = 6$ and $q = 10$ variables in the final model. This exercise was repeated,

with the difference being that increasingly less knot values were used for the linearisation of the log-likelihood objective function. Table 7.3 displays the objective function value obtained and time consumed during each round of model fitting, whereas Table 7.4 shows by how much the final solution has weakened alongside any potential improvements in model execution time.

Table 7.3: Model results for different values of k for the JUNKMAIL dataset

	$\log L$					Time (in seconds)				
k	$q = 6$	$q = 7$	$q = 8$	$q = 9$	$q = 10$	$q = 6$	$q = 7$	$q = 8$	$q = 9$	$q = 10$
401	-796.853	-763.082	-740.659	-720.951	-705.347	3012.08	5479.64	8104.39	11897.68	22255.88
161	-797.393	-764.197	-741.289	-722.038	-705.893	866.62	1615.34	2293.23	3696.87	6229.48
81	-799.669	-765.184	-744.089	-723.094	-707.086	366.51	610.10	859.92	1516.28	2308.04

Table 7.4: Deterioration in log-likelihood and improvements in run times for the JUNKMAIL dataset

	Deterioration in $\log L$					Reduction in run time				
k	$q = 6$	$q = 7$	$q = 8$	$q = 9$	$q = 10$	$q = 6$	$q = 7$	$q = 8$	$q = 9$	$q = 10$
401	-	-	-	-	-	-	-	-	-	-
161	-0.06%	-0.14%	-0.08%	-0.15%	-0.07%	-71.22%	-70.52%	-71.70%	-68.92%	-72.01%
81	-0.35%	-0.26%	-0.46%	-0.30%	-0.25%	-87.83%	-88.87%	-89.39%	-87.26%	-89.63%

From Tables 7.3 and 7.4 it can be seen that the use of a rougher grid (i.e. less grid values or a smaller k) results in a tremendous improvement in model execution time. Alternatively, no real deterioration in the final log-likelihood objective function values is evident. Finally, while not displayed here, note that Gini measures and accuracy rates similar to those in Table 6.7 were found for all of the aforementioned models that were fitted to the JUNKMAIL dataset, regardless of the size of the grid.

As is the case with the JUNKMAIL dataset, empirical evidence suggests that the user sacrifices very little when it comes to the quality of the final model when a less granular grid is utilised during the fitment of logistic regression models to the ONLINE NEWS POPULARITY dataset. However, significant gains can be made with respect to model run times when the number of grid values k is smaller. This is supported by results presented in Table 7.5 where the MILP formulation in (5.20)–(5.26) was tasked with fitting a 22-variable model for a variety of values specified for k.

Table 7.5: Model results for different values of k for the ONLINE NEWS POPULARITY dataset

k	$\log L$	Time (in seconds)	Train Gini	Test Gini	Train Accuracy	Test Accuracy
401	-6298.610	9610.88	40.92	37.60	64.87%	63.92%
161	-6320.899	3139.32	40.29	37.51	64.93%	63.80%
81	-6333.254	1364.88	40.71	37.29	65.09%	63.52%

By studying Table 7.5, notice that all of the models that were fitted to the ONLINE NEWS POPULARITY dataset obtained Gini values between 37 and 41. In turn, a Gini of 37 to 41

translates to an AUC measure of approximately 68 to 71. These measurements closely resemble the model evaluation metrics associated with a random forest model and gradient boosting machine that was fitted to the same set of data by Fernandes et al. (2015). The random forest obtained by the authors had an AUC of 72.7, while the gradient boosting machine achieved an AUC of 74.5. This suggests that a rudimentary classifier like the logistic regression MILP, which produces simpler models that are arguably more interpretable, has the ability to yield final models with similar predictive performance when compared to much more complicated machine learning algorithms, such as random forests and gradient boosting machines.

The SUPERCONDUCTIVITY DATA dataset is by far the largest real-world dataset considered within the best subset variable selection problem setting presented in this thesis. Not only does the training set consist of many rows ($n_1 = 6\ 000$ cases), but it contains 81 potential predictors of various scales, which is about twice the number of variables that best subset selection procedures are expected to handle efficiently. For this reason, all logistic regression models that were fitted to the SUPERCONDUCTIVITY data were allowed to execute for a maximum of 86 400 seconds, which translates to 24 hours. Table 7.6 summarises the results obtained during each round of model fitting when the logistic regression MILP was used to obtain 10-variable solutions.

Table 7.6: Model results for different values of k for the SUPERCONDUCTIVITY DATA dataset

k	$\log L$	Optimality Gap	Train Gini	Test Gini	Train Accuracy	Test Accuracy
401	-2038.729	22.57%	85.46	85.68	85.10%	84.49%
161	-1977.003	20.02%	86.49	86.67	85.20%	84.96%
81	-1955.179	20.37%	86.27	86.51	86.07%	86.04%
41	-1969.546	17.26%	87.33	87.49	86.57%	86.67%

Even though none of the models were able to conclude their execution within the specified time limit, Table 7.6 suggests that high quality solutions are still achievable even when model fitting is stopped early. Specifically, notice that improvements in the final log-likelihood objective function value and optimality gap are realised when the MILP formulation employs a grid that is more coarse. This is supported by consistently higher Gini and accuracy measures associated with lower k values. This can most likely be attributed to to the fact that the solver has to consider fewer constraints within the limited amount time that is made available for model fitting.

The results presented in this chapter indicate that the use of a less granular grid when performing best subset selection with the model in (5.20)–(5.26) entails many benefits. For moderately large datasets, such as the JUNKMAIL dataset and ONLINE NEWS POPULARITY dataset, considerably improved model run times can be achieved if the modeller is willing to accept an insignificant decay in the quality of the final solution. For much larger datasets, like

the SUPERCONDUCTIVITY DATA dataset, the use of a grid consisting of less knot values can still yield satisfactory models alongside potential improvements in the optimality gap and overall model fit, even when model execution terminates early.

Chapter 8

Conclusion and future work

8.1 Conclusion

Rapid advances in computing power and algorithmic execution in recent times have lead to a resurgence in interest in research directed towards the optimisation of linear models using exact approaches. Authors such as Bertsimas et al. (2016) present the formulation of linear regression models with best subset selection as mixed integer optimisation problems and demonstrate that satisfactory model solutions can be obtained for reasonably sized datasets within an acceptable amount of time. Sato et al. (2015) proposes a linearised approximation for the logistic loss function in binary logistic regression and subsequently facilitates variable selection via the use of a MILP formulation which selects the optimal model based on penalties such as the AIC and BIC. The resulting linearised model serves as an underestimator for the AIC/BIC and overestimator for the log-likelihood.

In Chapter 5, the estimation of the parameter vector in a logistic regression model using a linearised approximation of the log-likelihood function, was shown. The proposed linearised logistic regression model yields a solution that underestimates the maximum likelihood estimate. Experimental results indicate that the linearised model produces results that are comparable (or identical) to those obtained by models fitted using the iterative Newton-Raphson method. In some cases, the linearised model performed better than its iterative counterpart. The suggested model (without subset selection) yields solutions within a reasonable amount of time when the number of predictors, p, is in the 100's and the number of observations, n, is in the 1000's, noting that scalable solutions are still attainable even when n is notably larger at 20 000 records.

The proposed model was also extended to facilitate best subset selection via the introduction of cardinality parameters. The results presented in this thesis show that the linearised logistic regression model with best subset selection consistently produces parsimonious final models that are similar to or occasionally better than the benchmark. Furthermore, it is observed that a trade-off exists between accuracy and execution time when considering the number of

grid values, k, employed in the linearisation of the log-likelihood objective function. Empirical evidence provided in Chapter 7 show that the logistic regression MILP with best subset selection can produce optimal (or near-optimal) models within increasingly shorter computing times if a suitable value for k is specified. In fact, by shrinking k quite significantly, the modeller is able to obtain solutions considerably quicker without having to sacrifice much on the quality of the solution achieved. For substantially larger datasets, the use of less grid values still yields satisfactory logistic regression models, even when model execution is stopped early. Additionally, potential improvements in overall model fit and accuracy measures can be realised when k is reduced for sizable modelling sets.

Ultimately, the work presented in this thesis – which was accepted for publication (Venter and Terblanche, 2019) – suggests that the objectives that were outlined in Chapter 1 can be achieved with reasonable success. When the MILP formulation in (5.20)–(5.26) was used to fit logistic regression models to either simulated or real-world datasets, it was shown that the modeller is able to obtain parsimonious yet accurate models as a final solution. In addition, the logistic regression MILP provides a guarantee on the optimality of the resulting final model, which allows the modeller to evaluate the quality of solution. The proposed linearised model is also able to select variables for inclusion in the final model based on their ability to maximise the log-likelihood objective function as opposed to relying on p-values obtained from statistical tests, which are commonplace in more conventional approaches such as stepwise selection. Finally, it was shown in Chapter 7 that the MILP approach proposed in Chapter 5 is able to achieve satisfactory results within appropriate time frames, even when a significantly large dataset is considered. This demonstrates that viable solutions for hard optimisation problems, such as best subset selection, are attainable for increasingly larger sets of inputs – even for users with basic computing technology at their disposal.

8.2 Future work

In this section, a few open-ended topics with respect to the logistic regression MILP which may require additional investigation and research, are discussed. Firstly, consider the suggested approach on how to select an appropriate grid for the linearised model that was presented in Chapter 6. While the approach put forth for selecting suitable start and end points for the grid is relatively user-friendly and requires little mathematical rigour, alternative methodologies towards the specification of the grid can perhaps be explored.

Secondly, it is noted that the suggested model with best subset selection tends to find the best q-variable model and its corresponding parameter estimates fairly quickly, but requires a significant amount of time to prove optimality. In turn, this can result in increased execution times. However, this is not unique to the model suggested in (5.20)–(5.26) and has been docu-

mented for best subset selection applications in linear regression as well – see Bertsimas et al. (2016). Therefore, further consideration should be given to the structure of the MILP constraint matrix and the decomposition thereof in order to achieve additional algorithmic benefits and reduced model run times.

Next, the inclusion of additional sets of constraints in the MILP formulation in (5.20)–(5.26) which allow the modeller to have more control over the final model that is obtained, should be contemplated and tested. As part of their MIQP formulation for linear regression, Bertsimas and King (2016) develop and propose a variety of constraints that are imposed on the input variables, which in turn would require much more involvement from the user. The authors argue that the suggested model benefits from the strenghts inherent to both humans and machines. A human (in this case, the modeller) may be aware of certain structures that exist within the data and might have practical experience in and contextual knowledge of the problem being dealt with. Computers, on the other hand, have significantly more algorithmic prowess and computing abilities and can arrive at a solution much faster than any human. Ultimately, the authors believe that their approach provides modellers with a detailed, well-rounded methodology that enables the user to build high quality models while balancing the trade-off between his or her intuition and the modelling software's pursuit of the most optimal solution. Given this background, consider then the following MIQP linear regression formulation by Bertsimas and King (2016):

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\beta\|_2^2, \tag{8.1}$$

subject to

$$-M_{lL}z_l \leq \beta_l \leq M_{lU}z_l \quad \text{for } l = 1, \ldots, p, \tag{8.2}$$

$$z_l \in \{0, 1\} \quad \text{for } l = 1, \ldots, p, \tag{8.3}$$

$$\sum_{l=1}^{p} z_l \leq q, \tag{8.4}$$

$$z_i = \cdots = z_j \quad \forall z_i, \ldots, z_j \in S \quad \text{and } i \neq j, \tag{8.5}$$

$$z_i + z_j \leq 1 \quad \text{for } i \neq j, \tag{8.6}$$

$$z_l = 1, \tag{8.7}$$

$$\sum_{l \in T_l} z_l \leq 1. \tag{8.8}$$

The objective function in (8.1) and constraints in (8.2)–(8.4) are not new at this point and have already been presented in Chapter 5. However, the supplementary constraints in (8.5)–(8.8), which are referred to as selective sparsity constraints by Bertsimas and King (2016), add further depth to the model and will explained in more detail. The constraint in (8.5) is known

as a group sparsity constraint and allows the user to manage the model's ability to include variables that display a group structure. Specifically, the constraint will insure that if one variable from the set S is selected, all variables in S will subsequently be included in the model as well. Alternatively, if one variable is omitted from the resulting solution, all of the features in S will also be excluded. It should be noted that more than one set of predictors with a group structure can exist within the modelling set, which can be denoted as $S_1, S_2, \ldots, S_m$, for $m > 1$. In such problem settings, the MIP model formulation will contain m constraints of the form shown in (8.5). When a high degree of pairwise linear correlations between predictor variables are present within the data (which is often the case in real-world datasets), constraint (8.6) can be utilised to limit the number of correlated predictors in the final model. In fact, if the magnitude of the correlation $\rho_{i,j}$ that exists between two design matrix columns $\mathbf{X}_i$ and $\mathbf{X}_j$ is above an unacceptable cut-off level that is specified by the modeller, (8.6) will ensure that the final solution contains either variable X_i or X_j, but not both. Constraint (8.7) needs little elaboration and simply allows the modeller to include the l-th variable in the final model if he or she chooses to do so.

The constraint in (8.8) is somewhat intricate and requires a slightly more detailed discussion. A common task that is often carried out within predictive modelling settings relates to variable transformations, which involves the application of a variety of mathematical transformations and/or functions to raw predictor variable values in the hopes of finding a better relationship between the feature and the response or to deal with outliers. Suppose that the modeller has opted to apply a few transformations to the variable X_l and that these transformations – along with variable itself – is contained within the set T_l. A simple example would be $T_l = \{X_l, \log(X_l), X_l^{\frac{1}{2}}, X_l^2\}$. Constraint (8.8) will ensure that only one feature from the set T_l is considered for inclusion in the final model. Clearly, all of the elements in T_l will exhibit a high degree of dependency upon one another, since they are all based on the vector $\mathbf{X}_l$. It would therefore be ill-advised to include more than one of these features, which substantiates the necessity for (8.8).

The auxiliary constraints in (8.5)–(8.8) can easily be applied to the logistic regression MILP in (5.20)–(5.26) with very little effort. The use of these constraints might entail additional benefits, such as allowing the modeller to use subject matter expertise in his or her approach and to have more control over the resulting solution, which allows for a more transparent modelling process. Furthermore, it provides a foundation for sound modelling principles to be applied, which includes endeavours such as the removal of highly correlated or linearly dependent input variables. Given the aforementioned work done by Bertsimas and King (2016), insights might be obtained by applying the MILP model in (5.20)–(5.26) to both simulated and real-world datasets with the addition of constraints similar to those listed in (8.5)–(8.8).

Lastly, the properties of the logistic regression estimate vector $\hat{\beta}$ produced by the linearised

model should be explored to a greater extent. Specifically, the effect of the upper and lower bounds M_{lL} and M_{lU} in constraint (5.24) in (5.20)–(5.26) and the size of the grid should be investigated, as significantly lower upper bounds, higher lower bounds or a narrower grid has the potential to shrink fitted regression coefficients, which could yield biased estimates. In order to grasp the concept, let $\hat{\beta}^*$ be the estimated regression coefficient vector that produces a maximum value for the log-likelihood objective function when no constraints are imposed on the size of the estimated coefficients. Furthermore, suppose that a grid $[A,B]$ consisting of k knot values, with A and B the end-points of the grid, is used in (5.20)–(5.26). If a substantial proportion of the observations in the dataset are associated with $\mathbf{X}_i\hat{\beta}^*$ values that are significantly lower than A or significantly larger than B, for $i = 1, \ldots, n$, the linearised model is potentially at risk of producing estimates such that $|\hat{\beta}_l^{MILP}| < |\hat{\beta}_l^*|$ for $l = 1, \ldots, p$. The presence of shrunken regression estimates might be considered as unwanted at first, but the concept should not be rejected entirely. When penalised regression methods were discussed in Chapter 3, recall that lasso regression has the ability to yield much more stable models when biased estimates are used, which are obtained by shrinking the estimated regression parameters. While the benefits and drawbacks of shrunken coefficient estimates can be discussed at great lengths, the level of bias inherent to estimated parameters obtained by the MILP formulation remains unclear at this point and should be subjected to additional scrutiny.

References

Acquah, H. (2010). Comparison of Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) in selection of an asymmetric price relationship. *Journal of Development and Agricultural Economics*, 2(1):1–6.

Adejo, B. and Okutachi, A. (2012). On the Ellipsoid Method for Systems of Linear Inequalities. *Advances in Applied Science Research*, 3(5):3354–3361.

Ahmadi, A. (2016). ORF523 spring 2015 lecture notes, public teaching, Princeton University. `http://www.princeton.edu/~amirali/Public/Teaching/ORF523/S16/ORF523_S16_Lec7_gh.pdf`. Accessed on 2019-03-16.

Akaike, H. (1973). Information Theory and an Extension of the Maximum Likelihood Principle. In B. N. Petrov, and F. Csaki (Eds.). In *Proceedings of the 2nd International Symposium on Information Theory*, pages 267–281, Budapest.

Anderson, R. (2007). *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation*. Oxford University Press.

Bayen, A., Tomlin, C., Ye, Y., and Zhang, J. (2003). MILP formulation and polynomial time algorithm for an aircraft scheduling problem. In *Proceedings of the 42nd IEEE International Conference on Decision and Control*, Maui, HI, USA.

Beck, A. (2014). *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. Society for Industrial and Applied Mathematics (SIAM).

Berkson, J. (1944). Applications of the Logistic Function to Bio-Assay. *Journal of the American Statistical Association*, 39(227):357–365.

Bertsimas, D. and King, A. (2016). OR Forum - An Algorithmic Approach to Linear Regression. *Operations Research*, 64(1):2–16.

Bertsimas, D., King, A., and Mazumder, R. (2016). Best Subset Selection Via a Modern Optimization Lens. *The Annals of Statistics*, 44(2):813–852.

Bishop, Y., Fienberg, S., and Holland, P. (1975). *Discrete multivariate analysis: Theory and practice*. MIT Press, MA, Cambridge.

Bixby, R. (2012). A brief history of linear and mixed-integer programming computation. In *Documenta Mathematica, Extra Volume: Optimization Stories. 21st International Symposium on Mathematical Programming*, pages 107–121, Berlin, Germany, Europe.

Bliss, C. (1934). The Method of Probits. *Science*, 79(2037):38–39; 409–410.

Bliss, C. (1935). The Calculation of the Dosage-Mortality Curve (with appendix by R.A. Fisher). *Annals of Applied Biology*, 22(1):134–167.

Bolton, C. (2009). *Logistic regression and its application in credit scoring*. PhD thesis, University of Pretoria.

Bradley, S., Hax, A., and Magnanti, T. (1977). *Applied Mathematical Programming*. Addison-Wesley Publishing.

Brandimarte, P. (2006). *Numerical Methods in Finance and Economics - A MATLAB-Based Introduction*. Wiley and Sons, Inc.

Breheny, P. and Huang, J. (2011). Coordinate decent algorithms for nonconvex penalised regression, with applications to biological feature selection. *The Annals of Applied Statistics*, 5(1):232–253.

Breheny, P. and Huang, J. (2015). Group decent algorithms for nonconvex penalised linear and logistic regression models with grouped predictors. *Statistics and Computing*, 25(2):173–187.

Breiman, L. (1993). Better subset selection using non-negative garotte. University of California, Berkeley.

Breiman, L. (1995). Better Subset Regression Using the Nonnegative Garrote. *Technometrics*, 37(4):373–384.

Buhlmann, P. and van-de Geer, S. (2011). *Statistics for high dimensional data*. Springer.

Bursac, Z., Gauss, C., Williams, D., and Hosmer, D. (2008). Purposeful selection of variables in logistic regression. *Source Code for Biology and Medicine*, 1(17):17.

Constanza, M. and Afifi, A. (1979). Comparison of Stopping Rules in Forward Stepwise Discriminant Analysis. *Journal of the American Statistical Association*, 74(368):777–785.

Cooper, W., Henderson, A., and Charnes, A. (1953). *An Introduction to Linear Programming*. John Wiley and Sons, Inc. New York; Chapman and Hall Ltd. London.

Cox, D. (1969). *Analysis of binary data*. London: Chapman and Hall.

Cramer, J. (2002). The Origins of Logistic Regression - Working Paper No. 2002-119/4. Tinbergen Institute.

Dantzig, G. (1951). *Maximization of a linear function of variables subject to linear inequalities. T.C. Koopmans Ed: Activity Analysis of Production and Allocation*. John Wiley and Sons, Inc. New York; Chapman and Hall Ltd. London.

Dantzig, G. (1953a). Alternate Algorithm For The Revised Simplex Method Using A Product Form For The Inverse. RAND Report RM-1266. The RAND Corporation, Santa Monica, CA.

Dantzig, G. (1953b). Computational algorithm of the revised simplex method. RAND Report RM-1266. The RAND Corporation, Santa Monica, CA.

Dantzig, G. (1963). *Linear Programming and Extensions*. Princeton University Press, New Jersey.

Dantzig, G., Orden, A., and Wolfe, P. (1954). The Generalized Simplex Method for Minimizing A Linear Form Under Linear Inequality Restraints. RAND Report RM-1264. The RAND Corporation, Santa Monica, CA.

Dasgupta, S., Papadimitriou, C., and Vazirani, U. (2008). *Algorithms, 1st Ed.* McGraw-Hill Higher Eduction, Boston.

Del Pia, A., Dey, S., and Molinaro, M. (2017). Mixed-integer quadratic programming is in NP. *Mathematical Programming*, 162:225–240.

Derksen, S. and Keselman, H. (1992). Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables. *British Journal of Mathematical and Statistical Psychology*, 45(2):265–282.

Draper, N. and Smith, H. (1998). *Applied regression analysis. 3rd Ed.* Wiley, New York.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, 32(2):407–499.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalised likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1361.

Fernandes, K., Vinagre, P., and Cortez, P. (2015). A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. In *Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence*, Portugal.

Fourer, R. (2014). The Origins of a Practical Simplex Method. In *Proceedings of the AMPL INFORMS Annual Meeting, Session MB08 - Celebrating the Contributions of George Dantzig: Applications and Software*, pages 1–21, San Francisco, California, USA.

Friedman, J. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting. *The Annals of Statistics*, 28(2):337–407.

Friedman, J., Hastie, T., Tibshirani, R., and Hoeing, H. (2007). Pathwise coordinate optimization. *Annals of Applied Statistics*, 2(1):302–332.

Fu, W. (1998). Penalized Regression: The bridge Versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416.

Furnival, G. and Wilson, R. (1974). Regression by Leaps and Bounds. *Technometrics*, 16(4):499–511.

Gaddum, J. (1933). Report on Biological Standards III: Methods of Biological Assay Depending on Quantal Response. London Medical Research Council.

Gant, T. and Crowland, K. (2017). A Practical Guide to Getting Started with Propensity Scores. In *Proceedings of the 2017 SAS Global Forum*, pages 76 – 90, Orlando, Florida, USA.

Gatu, C. and Kontoghiorghes, E. (2003). Branch-and-Bound Algorithms for Computing BEST-Subset Regression Models. *Journal of Computational and Graphical Statistics*, 15(1):139–156.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, C., Mesirov, M., Coller, H., Loh, M., Downing, J., and Caligiuri, M. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):513–536.

Gomory, R. (1963). *Recent Advances in Mathematical Programming: An Algorithm for Integer Solutions to Linear Programs*. McGraw-Hill, New York.

Gurland, J., Lee, I., and Dahm, P. (1960). Polychotomous quantal response in biological assay. *Biometrics*, 16(3):382–398.

Gurobi Optimization (2017). Gurobi Optimization User Manual: Mixed Integer Programming (MIP) - A Primer on the Basics. `http://www.gurobi.com/resources/getting-started/mip-basics`. Accessed on 2018-03-26.

Hamidieh, K. (2018). A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354.

Harrel, F. (1997). Regression Modeling and Validation Strategies. School of Medicine, University of Virginia.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *Elements of Statistical Learning*. Springer Science and Business Media.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *Elements of Statistical Learning. Data Mining, Inference and Prediction. 2nd Ed. Springer Series in Statistics*. Springer New York.

Hastie, T., Tibshirani, R., and Tibshirani, R. (2017). Extended Comparisons of Best Subset Selection, Forward Stepwise Selection and the Lasso. ArXiv e-prints.

Hazimeh, H. and Mazumder, R. (2018). Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. Massachusetts Institute of Technology.

Heinze, G., Wallisch, C., and Dunkler, D. (2018). Variable selection - A review and recommendations for the practicing statistician. *Biometric Journal*, 60(3):431–449.

Hocking, R. and Leslie, R. (1967). Selection of the Best Subset in Regression Analysis. *Technometrics*, 9(4):531–540.

Hoerl, A. (1964). Ridge Analysis. Chemical Engineering Progress Symposium Series 60, p 67-77.

Hoerl, A. and Kennard, R. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67.

Hosmer, D. and Lemeshow, S. (2000). *Applied Logistic Regression. 2nd Ed.* Wiley and Sons, Inc.

Junger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleybank, W., Reinelt, G., Rinaldi, G., and Wolsey, L. (2009). *50 Years of Integer Programming 1958 - 2008*. Springer.

Kamiya, S., Miyashiro, R., and Takano, Y. (2019). Feature subset selection for the multinomial logit model via mixed-integer optimization. In *Proceedings of Machine Learning Research, PMLR*, volume 89, pages 1254–1263.

Karger, D. (2004). Advanced Algorithms: MIT course on Advanced Algorithms. `http://courses.csail.mit.edu/6.854/06/scribe/s15.pdf`. Accessed on 2019-03-16.

Kutner, M., Nachtsheim, C., Neter, J., and Li, W. (2005). *Applied Linear Statistical Models, 5th Ed.* McGraw-Hill.

Land, A. and Doig, A. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3):497–520.

Liao, H. (2010). A survey of variable selection methods in two Chinese epidemiology journals. *BMC Medical Research Methodology*, 10(1):87.

Luenberger, D. and Ye, Y. (2016). *Linear and Nonlinear Programming, 4th Ed. International Series in Operations Research and Management Science*. Springer International Publishing, Switzerland.

Lund, B. (2017). Logistic Model Selection with SAS PROC's LOGISTIC, HPLOGISTIC, HPGENSELECT - Paper AA02. In *Proceedings of the 2017 Midwest SAS Users Group Conference*, page 3, St. Louis, Missouri, USA.

Maldonado, S., Perez, J., Weber, R., and Labbe, M. (2014). Feature selection for Support Vector Machines via Mixed Integer Linear Programming. *Information Sciences*, 279(1):163 – 175.

Mantel, N. (1966). Models for complex contingency tables and polychotomous response curves. *Biometrics*, 22(1):83–110.

Mantel, N. (1970). Why stepdown procedures in variable selection. *Technometrics*, 12(3):621–625.

Mazumder, R., Friedman, J., and Hastie, T. (2011). SparseNet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138.

McFadden, D. (1973). *Conditional Logit Analysis of Qualitative Choice Behavior. In: Zare Ed., Frontiers in Econometrics*. Academic Press.

McKelvey, R. and Zavoina, W. (1975). A statistical model for the analysis of ordinal level dependent variables. *Journal of Mathematical Sociology*, 4(1):103–120.

Miller, A. (2002). *Subset selection in regression. 2nd Ed.* CRC Press: Washington; Chapman and Hall.

Mitchell, J. (2000). *Handbook of Applied Optimization: Branch-and-Cut Algorithms for Combinatorial Optimization Problems*. Oxford University Press.

Myers, J. and Forgy, E. (1963). The Development of Numerical Credit Evaluation Systems. *Journal of American Statistical Association*, 58(303):799–806.

Myers, R. (1990). *Classical and Modern Regression with Applications*. PWS-KENT.

Natarajan, B. (1995). Sparse Approximate Solutions to Linear Systems. *SIAM Journal on Computing*, 24(2):227–234.

Nesterov, Y. (2007). Gradient methods for minimizing composite objective function, technical report number 76. Centre for Operations Research and Econometrics (CORE), Catholic University of Louvain.

Nesterov, Y. and Nemirovski, A. (1994). *Interior-Point Polynomial Algorithms In Convex Programming*. SIAM Studies in Industrial and Applied Mathmatics.

Neter, J., Wasserman, W., and Kutner, M. (1983). *Applied Linear Regression Models*. R.D. Irwin.

Oghojafor, B., Mesike, G., Omoera, C., and Bakare, R. (2012). Modelling telecom customer attrition using logistic regression. *African Journal of Marketing Management*, 4(3):110–117.

Okeh, U. and Oyeka, I. (2013). Estimating the fisher's scoring matrix formula from a logistic model. *American Journal of Theoretical and Applied Statistics*, 2(6):221–227.

Orchard-Hays, W. (1968). *Advanced Linear-Programming Computing Techniques*. McGraw-Hill Book Company.

Papadimitriou, C. (1981). On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28(4):765–768.

Patetta, M. (2012). *Predictive Modelling Using Logistic Regression Course Notes. Course code LWPMLR93/PMLR93*. SAS Institute Inc.: Cary, N.C.

Patetta, M., Huber, M., and Nizam, A. (2009). *Categorical Data Analysis Using Logistic Regression Course Notes. Course code LWCDAL92/CDAL92*. SAS Institute Inc.: Cary, N.C.

Potts, W. and Patetta, M. (1999). *Logistic Regression Modelling*. SAS Institute Inc.: Cary, N.C.

Press, W., Flannery, B., Teukolsky, S., and Vetterling, W. (1992). *Numerical Principles in FORTRAN: The Art of Scientific Computing, 2nd Ed.* Cambridge University Press.

Rawlings, J. (1988). *Applied regression analysis: a research tool*. Wadsworth and Brooks; Cole Advanced Books and Software.

Reed, L. and Berkson, J. (1929). The Adaption of the Logistic Function to Experimental Data. *Journal of Physical Chemistry*, 33:760–779.

Rezac, M. and Rezac, F. (2011). How to measure the Quality of Credit Scoring Models. *Journal of Economics and Finance*, 61(5):486–507.

Ryan, T. (1997). *Modern Regression Methods*. Wiley, New York.

Sahin, Y. and Ekrem, D. (2011). Detecting credit card fraud by ANN and logistic regression. INISTA 2011 - 2011 International Symposium on Inovations in Intelligent Systems and Applications. 10.1109/INISTA.2011.5946108.

SAS Institute Inc. (2017). *SASHELP Data Sets*. SAS Institute Inc.: Cary, NC.

Sato, T., Takano, Y., Miyashiro, R., and Yoshise, A. (2015). Feature subset selection for logistic regression via mixed integer optimization. *Discussion Paper Series No. 1324, Department of Policy and Planning Sciences, University of Tsukuba*.

Schechtman, E. and Schechtman, G. (2016). The relationship between Gini terminology and the ROC curve. The Social Science Research Network (SSRN). https://ssrn.com/abstract=2739245. Accessed on 2020-02-14.

Schwartz, G. (1978). Estimating the Dimension of a Model. *Annals of Statistics*, 6(2):461–464.

Siddiq, N. (2006). *Credit Risk Scorecards - Developing and Implementing Intelligent Credit Scoring*. Wiley and Sons, Inc.

Smith, G. (2018). Step away from stepwise. *Journal of Big Data*, 5(1):32.

Smith, J. and Taskin, Z. (2007). *A Tutorial Guide to Mixed-Integer Programming Models and Solution Techniques. Optimization in medicine and biology*. Taylor and Francis, Auerbach Publications.

Stanev, V., Oses, C., Kusne, A. G., Rodriguez, E., Paglione, J., Curtarolo, S., and Takeuchi, I. (2018). Machine learning modeling of superconducting critical temperature. *Computational Materials*, 4(29).

Taylor, J. and Tibshirani, R. (2015). Statistical learning and selective inference. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 7629–7634.

Theil, H. (1969). A multinomial extension of the linear logit model. *International Economic Review*, 10(3):251–259.

Thompson, B. (1995). Stepwise regression and stepwise discriminant analysis need not apply here: a guidelines editorial. *Educational and Psychological Measurement*, 55(4):525–534.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society, Series B*, 58(1):267–288.

Venter, J. and Terblanche, S. (2019). Variable selection in logistic regression models through the application of exact mathematical programming. *South African Statistical Journal*. Accepted for publication in 2020.

Walsh, S. (2002). *Applying Data Mining Techniques Using Enterprise Miner Course Notes. Course code ADMT.* SAS Institute Inc.: Cary, N.C.

Whittingham, M., Stephens, P., Bradbury, R., and Freckleton, R. (2006). Why do we still use stepwise modelling in ecology and behaviour? *Journal of Animal Ecology*, 75(5):1182–1189.

Wilson, E. (1929). The logistic or autocatalytic grid. *Proceedings of the National Academy of Science*, 11(8):431–456.

Wilson, J. and Lorenz, K. (2015). *Modelling Binary Correlated Responses using SAS, SPPSS and R. ICSA Book Series in Statistics 9.* Springer International Publishing, Switzerland.

Winsor, C. (1932). A comparison of certain symmetrical growth curves. *Journal of the Washington Academy of Sciences*, 22(4):73–84.

Zhang, C. (2010). Nearly unbiased variable selection under the minimax concave penalty. *Annals of Statistics*, 38(2):894–942.

Zhang, S., Qian, H., Chen, W., and Zhang, Z. (2013). A Concave Conjugate Approach for Nonconvex Penalized Regression with the MCP Penalty. In *Proceedings of the twenty-seventh AAAI Conference on Artificial Intelligence*, pages 1027–1033, Bellevue, Washington, USA.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 67(2):301–320.

Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood. *Annals of Statistics*, 36(4):1509–1533.

Appendix A: The likelihood ratio test for logistic regression

Kutner et al. (2005) provides a detailed explanation of the likelihood ratio test conducted for logistic regression. In this test, the global null hypothesis stating that all regression parameters in the regression model are zero, is considered. The alternative states that at least one regression coefficient is significantly different from zero. First, let $\mathbf{Y} \in \{0, 1\}$ be a $n \times 1$ binary target vector and $\mathbf{X}$ an $n \times p$ design matrix containing the input variables, where n is the number of observations in the dataset and p is the number of candidate predictors. The posterior probabilities given by the full model, which considers all of the inputs, are then expressed as

$$\pi = \frac{1}{1 + \exp(-\mathbf{X}\beta_F)}, \tag{A.1}$$

where

$$\mathbf{X}\beta_F = \beta_0 + \beta_1\mathbf{X}_1 + \cdots + \beta_p\mathbf{X}_p.$$

Let $L(\beta_F)$ be the value of the maximum likelihood estimate obtained for the full model by evaluating the maximum likelihood function at $\hat{\beta}_F$, where $\hat{\beta}_F$ are the estimates for the parameter vector β_F in the full model in (A.1). Furthermore, suppose that the modeller would like to test k regression coefficients. If the model is rearranged such that the last k parameters are the ones being tested, the hypothesis is given by:

$$H_0 : \beta_{p-k+1} = \beta_{p-k+2} = \cdots = \beta_p = 0$$

vs

$$H_a : \text{ at least one } \beta_l \text{ in } H_0 \text{ is not zero, for } l = p - k + 1, \ldots, k$$

Next, let the vector of predicted probabilities yielded by the reduced model be given by

$$\pi = \frac{1}{1 + \exp(-\mathbf{X}\beta_R)},$$

where

$$\mathbf{X}\beta_R = \beta_0 + \beta_1\mathbf{X}_1 + \cdots + \beta_{p-k}\mathbf{X}_{p-k}.$$

Suppose that $\hat{\beta}_R$ is the estimated coefficient vector for β_R. By evaluating the maximum likelihood function at $\hat{\beta}_R$, the value of the MLE is obtained for the reduced model containing $p-k$ inputs, denoted by $L(\beta_R)$. Notice that $L(\beta_R)$ can never be more than $L(\beta_F)$. The test statistic for the likelihood ratio test is then calculated as

$$G = -2\log\left[\frac{L(\beta_R)}{L(\beta_F)}\right] = -2[\log L(\beta_R) - \log L(\beta_F)].$$

If the sample size n is sufficiently large enough, the statistic G will follow a χ^2 distribution with k degrees of freedom. The degrees of freedom follow from $df_R - df_F = (n-(p-k)) - (n-p) = k$. Consequently, H_0 is rejected when $G > \chi^2(1-\alpha, k)$ for some significance level α and the modeller can conclude that at least one of regression coefficients associated with the k variables is non-zero.

Appendix B: The Receiver Operator Characteristic curve

Let π_i be the estimated probability for the i-th observation produced by a binary model $f(\mathbf{X}, \beta)$ that has been fitted to the data. Note that $f(\mathbf{X}, \beta)$ does not necessarily have to be a logistic regression model, but can be any predictor that produces probabilities as output, such as a decision tree, random forest or neural network. Next, assume that all observations with $Y_i = 0$ are referred to as "goods" while all observations with $Y_i = 1$ are known as "bads". Finally, let π_{co} be an arbitrary cut-off probability between zero and one. Consider then the cumulative proportion of goods and cumulative proportion of bads that have a predicted probability of π_{co} or less, denoted by $CG_{\pi_{co}}$ and $CB_{\pi_{co}}$, respectively. If $f(\mathbf{X}, \beta)$ is a sufficiently predictive and accurate model, then a higher proportion of goods and lower proportion of bads will have $\pi_i \leq \pi_{co}$ if π_{co} is relatively low. As π_{co} increases, more and more goods are captured, but the number of bad cases that are added to $CB_{\pi_{co}}$ remain relatively low. A certain point in the probability range denoted by π^*_{co}, is then reached, where the number of bads contributing to $CB_{\pi_{co}}$ start to increase rapidly for $\pi_{co} > \pi^*_{co}$. At this point, the majority of goods would have been captured, with the number of goods adding to $CG_{\pi_{co}}$ declining at an accelerated pace from π^*_{co} onwards. Figure B.1 serves as a visual representation of the ROC curve, where the blue line is based on table B.1 below it.

Notice the clear turning point on the blue ROC curve where π^*_{co} resides (shown by the dashed lines). At this point, most of the goods have been captured and the number of bads adding to $CB_{\pi_{co}}$ start to increase. However, such a clear turning point may not always exist within an ROC curve. In some cases, the curve may have a more rounded shape, where the number of goods added to $CG_{\pi_{co}}$ decrease at a much steadier pace, while $CB_{\pi_{co}}$ increases gradually. Such an ROC curve is depicted by the orange line. The red line is referred to as the baseline model and is illustrated graphically on all ROC curves as a line at a 45° angle. The baseline model is often known as a random model and represents the level of accuracy that the modeller can expect if classifications were made based solely on chance. This would be equivalent to assigning a probability to the i-th observation using a random number generator.

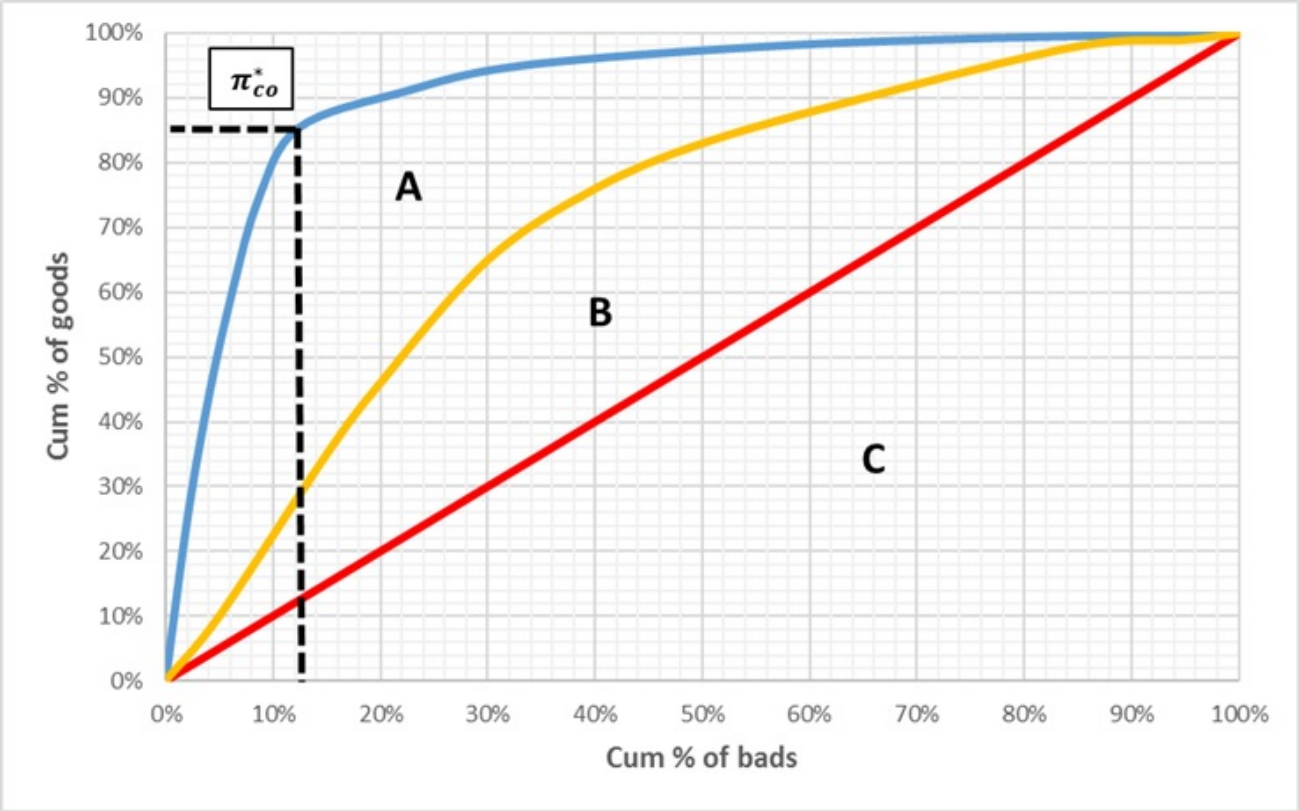


Figure B.1: ROC curve for two separate models

Table B.1: Calculating the ROC curve

π_{co}	# goods	# bads	% goods	% bads	cum % goods	cum % bads
0	0	0	0%	0%	0%	0%
0.1	9000	187	25%	2%	25%	2%
0.2	7000	190	19%	2%	44%	4%
0.3	6000	210	17%	2%	61%	6%
0.4	5000	220	14%	3%	74%	9%
0.5	4000	350	11%	4%	85%	12%
0.6	2000	900	6%	10%	91%	22%
0.7	1500	1050	4%	11%	95%	33%
0.8	1080	2150	3%	23%	98%	56%
0.9	470	2000	1%	21%	99%	78%
1	261	2100	1%	22%	100%	100%

Let $f_1(\mathbf{X}, \beta)$ be the model associated with the blue ROC curve, while $f_2(\mathbf{X}, \beta)$ is the model that is represented by the orange ROC curve. For $f_1(\mathbf{X}, \beta)$ the AUC is equal to the area covered by regions A, B and C, i.e. $AUC_1 = A + B + C$. Alternatively, regions B and C contribute to the AUC of $f_2(\mathbf{X}, \beta)$, where $AUC_2 = B + C$.

Rearranging (3.10) in Chapter 3 shows that the Gini is calculated as

$$Gini = \frac{AUC - 1}{2}.$$

In fact, the Gini is simply equal to the area under the curve that does not include the region C shown in Figure B.1. The Gini can therefore be interpreted as a metric that measures the incremental increase in ranking ability achieved by using a model that does not make decisions based on chance.

In general, higher AUC and/or Gini values suggest a superior model. Such models are considered to "rank" better due to the assignment of lower probabilities to non-event cases and higher probabilities to event cases.

Appendix C: Branch-and-bound methods for best subset selection

Two prominent branch-and-bound methods suggested for performing best subset selection when fitting regression models to data include the *Leaps and Bounds* or LBA method of Furnival and Wilson (1974) and the branch-and-bound algorithm of Gatu and Kontoghiorghes (2003), known as the BBA. Specifically, the software package SAS makes use of the LBA of Furnival and Wilson (1974) when best subset selection is carried out during model training. Both methods are executed by generating a regression tree, with each node in the tree storing the output of regression models that contain a subset of all the potential variables which are considered during the model fitting process. The LBA method generates two trees, where each tree consists of 2^{p-1} nodes and where p is the total number of variables considered. During each iteration of the LBA approach, the inverse of the moments matrix (which consists of the design matrix $\mathbf{X}$ and the output vector $\mathbf{Y}$ in the model $\mathbf{Y} = \mathbf{X}\beta + \varepsilon$) is computed and pivoting by means of Gaussian elimination is applied to a bounds tree. In both the LBA and BBA methods, the error sums of squares or SSE is utlised as an evaluation metric in each node of the tree (for logistic regression, the log-likelihood is used). Additionally, the SSE also serves as lower and upper bounds throughout the tree.

The LBA approach generates a bounds tree which provides half of the models that do not include the last variable. Gaussian elimination is then used to obtain the next node in the tree. Ultimately, the goal is to obtain tighter upper and lower bounds during early stages of the algorithm, meaning that branching needs to occur on fewer subsequent nodes in the tree. The second tree – known as the regression tree – contains the other half of the models that include the last variable. Models are then computed by moving from one node to the next, where the matrix inverse is calculated during each iteration.

The BBA approach also makes use of a regression tree. But, instead of using matrix inverses, it utilises the QR decomposition of the exogenous data matrix $\mathbf{X}$ and the dependent vector $\mathbf{Y}$. Empirical studies have shown that the BBA is superior to the LBA when considering tractability and computational efficiency. For this reason (also taking into account the close similarities that exist between the two approaches) the BBA method will be expained in more detail next.

At any given point in the BBA regression tree, each node consists of regression models that contain at most $p-1$ predictors. Every node is represented by a tuple (V, k), where V is the set of indices of the variables included in the regression models for the particular node in question and k indicates that the children of the node will include the first k inputs (note that in this instance, the value k does not refer to the number of grid values used in the linearised logistic regression model presented in Chapter 5). Figure C.1 provides a graphical example of steps in a regression tree.

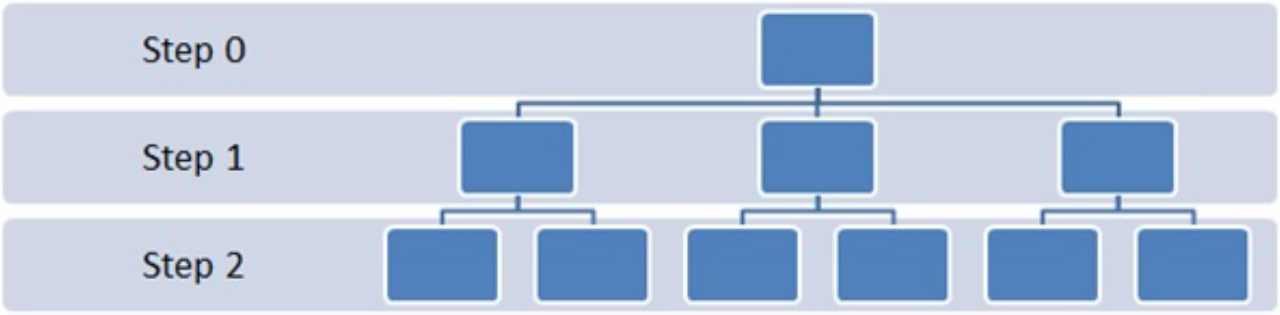


Figure C.1: Steps of a regression tree

At each step in the tree, moving from left to right, every regression model in a node will include the first k variables for the particular node in question, with $k = 0, 1, \ldots, V-1$. When $k = 0$, it implies that all combinations of variables are included in the node. Assuming that i is the number of the child node being considered in the current step and k is the k-value from the previous node, the $(i+k)$-th variable from the previous node is dropped in the current node. Figure C.2 serves as a graphical summary of the aforementioned regression tree process for a simple regression problem consisting of five potential predictors. In Figure C.2, the values above the line in each node represent the indices of the variables to be considered during regression model fitting for that particular node. The k-value for each node is also given. The values below the line represent the variable combinations that feature in the fitted regression models that belong to the node being considered. Lastly, i serves as the node's number.

The regression tree in Figure C.2 can be interpreted as follows:

- At step 0 the algorithm starts with $k = 0$, meaning that a regression model is produced which includes all the variables in this node, namely variables 1, 2, 3, 4 and 5. The node also yields other regression models, where the last variable from the previous model is omitted at each subsequent iteration. Four alternative regression models are obtained, which include a model with variables 1, 2, 3 and 4, a model with variables 1, 2 and 3, a model with variables 1 and 2 and lastly a model that includes variable 1. The node at step 0 is referred to as the root node (the start of the tree).

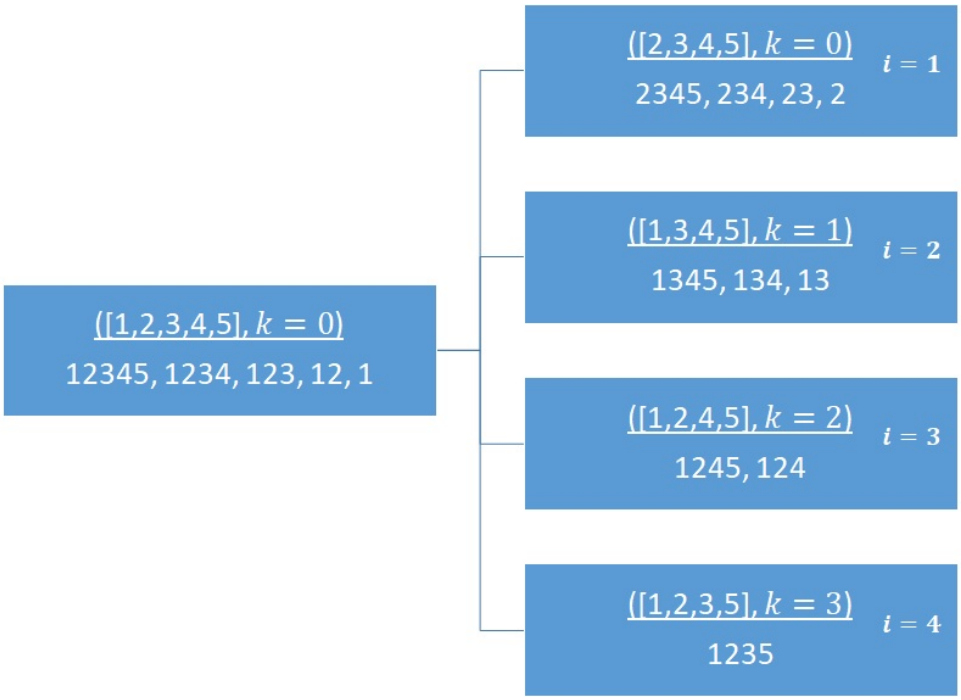


Figure C.2: Regression tree example

- At $i = 1$, the LBA alogorithm considers the first child node. Note that variable 1 has been deleted since $i = 1$. Given that $k = 0$, it is known that the first model will include all of the variables in this node. All subsequent models will include all of the variables of the latest model obtained in this node, except for the last variable.

- At $i = 2$, the second child node is produced. The second variable is not considered in this node. It can be seen that $k = 1$, which means that all models in this node will contain variable combinations that include the first variable in the subset (in this case, variable 1).

- The third child node is considered at $i = 3$ where the third variable has been removed. Note that $k = 2$, implying that all models in this node will include the first 2 variables (in this case, variables 1 and 2) in their variable combinations. As in the previous child nodes, the last variable from the model obtained in the previous iteration is dropped from the list of inputs considered each time when a model is fitted.

- Finally, at $i = 4$, the fourth child node is obtained. This time, the fourth variable has been deleted. Since $k = 3$, all models in this node will include variables 1, 2 and 3 in their variable combinations. Given that variable 5 is the only remaining predictor, only one model consisting of the required combination is produced.

While the nodes at $i = 1$, $i = 2$ and $i = 3$ will have subsequent child nodes where the same methodology shown in Figure C.2 is applied, it should be noted that the fourth node at $i = 4$

will have no child node, since only one single model consisting of variables 1, 2, 3 and 5 is yielded by this node, i.e. no other models are obtained that can be dissected further.

Both the LDA and BBA use the following principle when cutting nodes from the tree in order to reduce execution time, which is expressed as

$$SSE(V_1) \geq SSE(V_2), \tag{C.1}$$

where V_1 and V_2 are subsets of the p potential variables available for consideration and $V_1 \subseteq V_2$, meaning that V_1 includes exactly the same or less of the inputs contained in V_2. The same applies in logistic regression, noting that

$$\log L(V_1) \leq \log L(V_2). \tag{C.2}$$

In (C.2) the sign is reversed, since the log-likelihood should be maximised in logistic regression. Alternatively, the minimisation of the SSE yields the optimal linear regression model.

Suppose that the BBA is faced with a linear regression problem where $\mathbf{Y} \in \mathbb{R}$. Given the assumptions made in (C.1) and (C.2), the algorithm then proceeds to cut child nodes by using the following approach:

1. Let $r_l^{(g)}$ be the SSE for the best l-variable model having the smallest SSE of all l-variable models after g nodes of the regression tree have been considered so far, with $l = 1, \ldots, p$.

2. Let $b_v \equiv \min\ SSE(V)$, where V is the subset of all variables considered in the current node. For example, consider the child node in Figure C.3, where the SSE of each model is given in brackets below their respective variable combinations. In this node, the variables 3, 4 and 5 are considered and three models are obtained, namely a model consisting of variables 3, 4 and 5, a model with variables 3 and 4 and finally a model containing only variable 3. In this case, $b_v = 720$.

3. If $b_v \geq r_l^{(g)}$, then all subsequent branches of the current child node can be cut, since it is no longer necessary to compute them. For example, suppose that $r_5 = 668$, $r_4 = 615$, $r_3 = 597$, $r_2 = 592$ and $r_1 = 548$ are optimal values that apply at the current iteration of the BBA alogorithm. The node in Figure C.3 can only produce child nodes that contain 1- and 2-variable models if the BBA approach that is partially shown in Figure C.2, is followed. By referring to (C.1), it can be assumed that the SSE of any 1- and 2-variable model originating from the node in Figure C.3 will be greater than or equal to $b_v = 720$. Since $r_2 = 592$ and $r_1 = 548$, it is clear that none of the child nodes branching from this node can improve the SSE, given that $b_v \geq r_2$ and $b_v \geq r_1$. Ultimately, it implies that the child nodes of this particular node can be discarded and do not have to be evaluated.

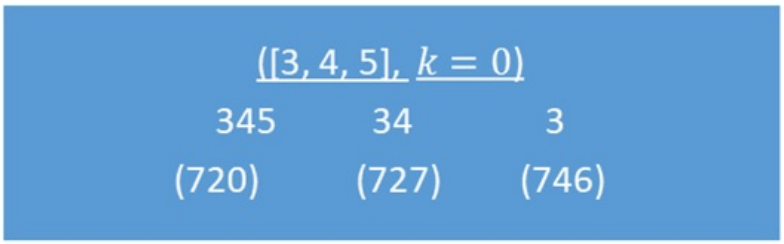


Figure C.3: Child node example

Gatu and Kontoghiorghes (2003) propose various ways for improving the speed of the BBA algorithm, some of which are discussed below. Firstly, the authors introduce the BBA-1 algorithm, which results in more nodes being cut from the tree by pre-ordering variables in the initial set such that

$$SSE(Drop(V,1)) \geq SSE(Drop(V,2)) \geq \cdots \geq SSE(Drop(V,p-1)),$$

where

$$Drop(V,i) = [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_p],$$

noting that v_i denotes the indices of the variables considered. The BBA-1 method suggests that the most significant inputs which are deemed to be more predictive should be placed first in the queue of variables being evaluated, while predictors that are considered to be less important should appear last.

In addtion, the authors also develop a derivation of the BBA method known as the BBA-2 approach. The BBA-2 algorithm may yield further computational improvements over the vanilla BBA method by processing the node with the smallest SSE (or largest log-likelihood in the case of logistic regression) first at each step. This means that tighter bounds for the SSE can be obtained earlier on during model execution, resulting in more nodes being cut during the initial stages of the algorithm. In notation, if $(V_1, k_1), \ldots, (V_d, k_d)$ denotes d possible nodes in the current step of the regression tree, with $0 \leq d \leq 2^{p-1}$, then the node corresponding to $\min\{SSE(V_1), \ldots, SSE(V_d)\}$ is processed first.

Lastly, a heuristic version of the BBA approach, called the HBBA, is suggested for high-dimensional problems. The HBBA uses one additional parameter known as a tolerance parameter τ, with $0 \leq \tau \leq 1$. Recall that a node in the regression tree is cut when the minimum SSE of the node is greater than or equal to $r_l^{(g)}$ when an exhaustive version of the BBA is applied. Alternatively, the authors suggest cutting a node when $SSE \geq (1-\tau)r_l^{(g)}$. In general, as τ tends to zero, the modeller has a better chance of finding an optimal solution. When $\tau = 0$, the exhaustive BBA is executed. Notice that the introduction of τ is strikingly similar to the concept of an optimality gap for mixed integer programming models, which was introduced in

Chapter 4. The smaller the optimality gap, the closer the model is to the optimal solution. Similarly, a smaller τ delivers a solution that is closer to optimality. Gatu and Kontoghiorghes (2003) include a proof showing that the relative error of the solution yielded by the HBBA will never be more than the tolerance parameter τ. That is, if ϵ represents the relative error of a solution found by means of HBBA and $\tau = 0.05$, then $\epsilon \leq 5\%$.

After conducting many experimental studies on both real-world and simulated datasets, Gatu and Kontoghiorghes (2003) concluded the following:

- BBA with pre-ordering, namely BBA-1, is faster than the heuristic HBBA when the tolerance parameter τ is not very large.
- A heuristic version of BBA-1, HBBA-1, is much faster than HBBA. Therefore, HBBA-1 can use smaller tolerance parameters while simultaneously deriving models that are closer to optimality.
- HBBA-1 finds models that are closer to optimality as opposed to HBBA when the same tolerance parameter τ is used.
- As the number of potential variables under consideration for inclusion in the final model increases, the tolerance parameter τ which is applied in the heuristic approach needs to be reduced in order to obtain correct models.
- The relative error ϵ can be used effectively when considering models and their solutions. That is, the modeller will always know that the relative error of the solution is less than or equal to the chosen tolerance parameter τ.
- Multicollinearity amongst predictors will exert an influence on execution time. This is due to the fact that more non-zero predictors in the true model will result in more branches being cut, which speeds up execution. Therefore, the impact of data quality on the efficiency of the BBA algorithm must be investigated.