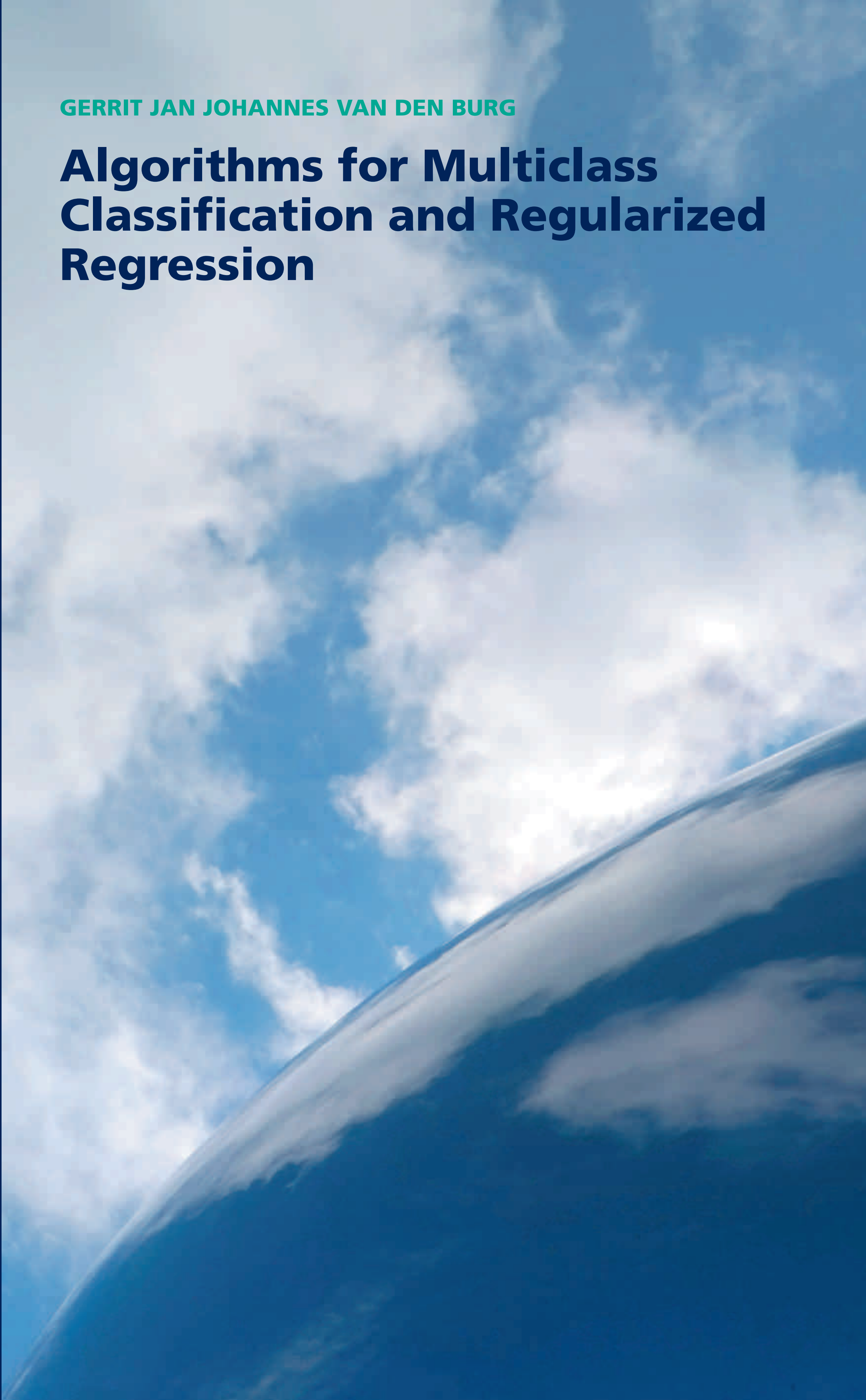
GERRIT JAN JOHANNES VAN DEN BURG

Algorithms for Multiclass Classification and Regularized Regression



Algorithms for Multiclass Classification and Regularized Regression

Algorithms for Multiclass Classification and Regularized Regression

Algoritmes voor classificatie en regressie

Thesis

to obtain the degree of Doctor from the
Erasmus University Rotterdam
by command of the
rector magnificus

Prof.dr. H.A.P. Pols

and in accordance with the decision of the Doctorate Board.

The public defence shall be held on
Friday January 12, 2018 at 09:30 hrs

by

GERRIT JAN JOHANNES VAN DEN BURG
born in 's-Gravenhage.

Erasmus University Rotterdam

Doctoral Committee

Promotor:	Prof.dr. P.J.F. Groenen
Other members:	Prof.dr. D. Fok
	Prof.dr. A.O. Hero
	Prof.dr. H.A.L. Kiers
Copromotor:	dr. A. Alfons

Erasmus Research Institute of Management – ERIM
The joint research institute of the Rotterdam School of Management (RSM)
and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam.
Internet: https://www.erim.eur.nl.

ERIM Electronic Series Portal: https://repub.eur.nl

ERIM PhD Series in Research in Management, 442
ERIM reference number: EPS-2017-442-MKT
ISBN 978-90-5892-499-5
© 2017, G.J.J. van den Burg

Design: G.J.J. van den Burg.

This publication (cover and interior) is printed by Tuijtel on recycled paper, BalanceSilk®.
The ink used is produced from renewable resources and alcohol free fountain solution.
Certifications for the paper and the printing production process: Recycle, EU Ecolabel,
FSC®, ISO14001.
More info: https://www.tuijtel.com.

MIX
Paper from responsible sources
FSC® C007225
FSC www.fsc.org

EU Ecolabel
www.ecolabel.eu

The cost of perfection is infinite.
– C.G.P. Grey

Acknowledgements

The road to this dissertation started about seven years ago when I attended a lecture on Multivariate Statistics, taught by prof. Patrick Groenen. During this lecture the “support vector machine” classification method was introduced and I remember that I was immediately very excited about it. A few weeks later, I emailed prof. Groenen to ask whether it would be possible to start a thesis project related to these “SVMs”. Luckily for me, this was possible and we soon started working on a project that would eventually become my MSc thesis and, after some more work, Chapter 2 of this dissertation.

After finishing my two MSc theses, I took some time to travel before starting my PhD at Erasmus University Rotterdam in December 2012, with Patrick Groenen as my doctoral advisor. The first task of my PhD was to improve my MSc thesis on multiclass SVMs to such a level that it could be submitted to a top academic journal. This involved a lot more research, programming, and testing but eventually we managed to submit the paper in December 2014 to the Journal of Machine Learning Research, a respected journal in the machine learning field. After a few rounds of revision, the paper was accepted and it got published in December 2016.

While the first paper was under review, I started a second project that focused on regularized regression (Chapter 4 of this dissertation). For this project Patrick and I teamed up with Andreas Alfons, who had extensive experience with robust statistics and regularized methods. This collaboration really helped improve the paper and take it to the next level. This paper was submitted to a machine learning conference in 2015 but was unfortunately rejected for various reasons. In the end, we managed to extend the work and make the method much more general, as can be seen in Chapter 5. I am confident that we will be able to submit this work to a good statistics journal in the near future.

In the winter of 2016 I had the wonderful opportunity to visit the University of Michigan in Ann Arbor for three months to join the research group of prof. Alfred Hero at the EECS department. It was quite the adventure living in the U.S. for three months (especially when cycling to the university through a thick layer of snow!). More importantly however, it was an incredible learning experience to join a different research group for a little while and to work with the people in the group. I am also very happy that this visit resulted in an actual research project that grew out to become Chapter 3 of this dissertation.

As the above paragraphs already highlight, I am indebted to a great number of people who contributed in one way or another to my time as a PhD candidate. First of all, I would like to thank Patrick Groenen for taking me on as an MSc student and as a PhD student and for giving me many opportunities to develop myself both as an independent researcher and as a teacher. I've enjoyed and learned a lot from our collaboration over the years. I especially admired your way of looking at a problem from a more geometric point of view, which often helped us to gain a lot more intuition about a problem. I certainly hope that even though my PhD ended, our collaboration will not.

I would also like to thank Andreas Alfons for being my co-advisor for the last few years of my PhD. Your keen eye for detail and your way of looking at the material has greatly improved Chapters 4 and 5 of this dissertation. Thanks also to prof. Alfred Hero for giving me the opportunity to visit the University of Michigan twice to work on a joined research project and for being part of my doctoral committee. I also want to extend my gratitude to all the members of my doctoral committee for taking the time to read this dissertation and for coming to Rotterdam for the defense.

During my time as a PhD student, I had the pleasure of sharing my office with two great colleagues. Niels Rietveld, thank you for welcoming me to the university and for showing me the ropes during the first few months. I'll never forget the first few days of my PhD when we worked on the window sill because the new desks hadn't arrived yet! Thomas Visser, thanks for being a nice office mate as well; despite our very different research topics I think we still learned from each other. Besides these two actual office mates, I would also like to thank Alexandra Rusu for being a quasi office mate two doors down the hall – thanks for the friendship and the many "interruptions" during the day that made the life of a PhD student looking at his computer screen all day much less boring!

There are two colleagues from the statistics department that I would like to thank here as well. Ronald de Vlaming, thank you for being a fellow statistician who, like me, wasn't afraid to fill a whole whiteboard with complicated formulas and then spend an evening discussing it. Pieter Schoonees, thank you for under-

standing what I was working on and for always being there to bounce some ideas off of. I am also grateful to both of you for the many fun moments at statistics conferences throughout the years.

In no particular order I would also like to thank my fellow PhD students from the Econometric Institute for the many lunches, stimulating conversations, and occasional drinks: Bruno, Bart, Francine, Myrthe, Tom, Sander, Didier, Koen, Aiste, Zarah, Anoek, Kevin, Judith, Charlie, and Rutger. I would like to especially thank Paul for the great collaboration on the programming course. I'm also grateful to Remy Spliet for joining me in a side project on classroom assignment from which I learned a lot, even though it didn't work out as well as we had hoped.

Being a PhD student at ERIM means you also get plenty of opportunities to meet colleagues from other departments. This was always a nice experience because it broadens your horizon a bit in terms of research done at the university, and often illustrates that the PhD experience is similar regardless of the topic of your PhD. For one year during my PhD I had the pleasure of joining the so-called "ERIM PhD Council", a sort of student body that exists as a layer between the ERIM staff and the PhD students. Together with Paul Wiegmann, Giorgio Touburg, and Eugina Leung, I had fun organizing academic and non-academic events for PhDs and I would like to thank them for joining this committee with me.

During my research visit at the University of Michigan, I had the pleasure of meeting a number of people who welcomed me to the university and made my stay in the U.S. an enjoyable one. I would especially like to thank Tianpei Xie for being a great office mate during my stay at the group, for showing me how everything worked there, and for making me feel like I was part of the group from the beginning. I would also like to thank Salimeh Sekeh, Oskar Singer, and the other members of the Hero group for making me feel welcome and for the stimulating conversations about machine learning. Besides the academic colleagues, I am also grateful to Maarten Roos, my serendipitous Dutch house mate in Ann Arbor, for showing me the social side of the town, and for making me feel welcome in the house.

Besides the many academic colleagues, I am also grateful for the friends in my life that exist outside academia. First of all, I would like to thank Raymon Mostafa and Gerard de Witt not only for being my friends for (almost) as long as I can remember, but also for being my paranymphs on the day of the defense (and for being so excited about it!). You two have always kept me grounded and reminded me that there is more to life than research. Thanks also to Tridib Das for being my friend throughout the PhD and for the many rooftop parties, silent discos, and Hawaiian-themed Christmas drinks that I had no idea Rotterdam had to offer! I would also like to thank Tim Baart for being a good friend for many years before and during the PhD, for helping me not to take academia too seriously, and for showing me that, yes, that other whiskey is good to try too. I would also like to thank Annemieke, Seffie, Anna, Smriti, Amanda, Teo, Oana, Leo, Robert, Alex, and the “SSL Culinair” group for their friendship and the various dinners and drinks throughout the years. Thanks also to C.G.P. Grey and dr. Brady Haran from Hello Internet for having me laughing all the way to work.

Besides colleagues and friends, there is one more group of people to thank: my family. I would like to thank my parents, Gerrie and Willem, for supporting me throughout the many years of studying as well as during my PhD. You have always believed in me and encouraged me even though I may not always have been able to explain what I was working on. Thanks also to Siepke and Diederik for being a great big sister and a great little brother, and to Jos for the “gezelligheid”. I am very grateful for the bond we have as a family. I am also grateful to Jeannet van den Burg for her warmth, energy, and wonderful humor.

That brings me to the final person to thank. Laura, meeting you has been one of the highlights of my PhD. Thank you for the many wonderful trips together all over the world, but especially to Romania where your family has always warmly welcomed me (even though I didn’t speak the language at all). Thank you for always being interested in my research and for letting me try to explain it even when I didn’t fully understand it myself, and thanks for being there for me when it really didn’t seem to work out. You always believe in me and for that I am very grateful. I look forward to many more lovely years together.

Gertjan van den Burg
Rotterdam, 2017

Table of Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

This dissertation presents several new algorithms for multiclass classification and regularized regression. Multiclass classification and regularized regression are both examples of so-called supervised learning techniques. Supervised learning is itself a subfield of machine learning and statistics. In the following paragraphs a broad introduction will be given to machine learning, supervised learning, classification problems, and regression problems. The individual chapters of this dissertation will be introduced in greater detail in the next section.

In the broad fields of machine learning and statistics there are various algorithms for all kinds of pattern recognition problems. The usual goal is to explain a phenomenon or predict an outcome, given the available data. Thus, the purpose of these machine learning methods is to discover an underlying relationship or predict a future event, as well as possible. When developing new algorithms, the goal is therefore to perform better than existing methods on some metric. Typically, such metrics measure how well a pattern is explained or how well an algorithm can predict unknown outcomes. Other important metrics are computation time and, perhaps more importantly, the ease of understanding an algorithm. The algorithms presented in this dissertation aim to advance the state of the art on at least some of these metrics.

All algorithms presented in this dissertation are supervised learning algorithms. This class of algorithms extracts a relationship between an observed outcome and available explanatory variables, with the goal of investigating the obtained relationship itself or to use it to predict the outcome of observations for which this is unknown (or both). Problems where it is desired to uncover such a relationship are ubiquitous in practice, with applications in econometrics, economics, finance, marketing, physics, chemistry, medicine, biology, psychology, sociology, and beyond. Within the class of supervised learning algorithms it is possible to make a distinction between problems where the outcome variable belongs to a limited set of classes and problems where it is continuous. The first type of problems are known as classification problems and the second type as regression problems. In this dissertation algorithms for both types of problems are presented.

Classification problems are typically prediction problems where the outcome variable belongs to one of several classes. For example, a spam filter on your computer predicts if an email belongs to the "spam" or "non-spam" class. Similarly, a bank may want to predict which customers are likely to default on a loan and a doctor may want to predict which disease a patient has based on historical records or blood analysis. In each of these examples it is assumed that there is some set of data with observations where the true class label is known. The goal of a classification algorithm is to identify patterns from this dataset in order to predict the class label of future observations. When more than two possible class labels are available, the problem is called a multiclass classification problem. The first part of this dissertation involves algorithms for multiclass classification problems.

In regression problems the outcome variable is continuous and the goal is to understand the relationship between the outcome variable and some explanatory variables and to predict the outcome variable for instances where it is unknown. For example, a university may be interested in how student grades depend on characteristics of the students such as class attendance, age, gender, study habits, and other variables. In this case, a regression analysis can be done to determine which variables have a significant influence on the grade of the student. When there is a large number of variables, a researcher may be interested in a sparse model in which the number of variables that are included in the model is limited. This is a form of *regularization*. In regularized models additional constraints are placed on the allowed solutions with the goal of achieving sparsity, encouraging simpler models, or limiting the size of the coefficients in the solution. Algorithms for regularized regression are the focus of the second part of this dissertation.

1.1 Overview of Chapters

As mentioned above, there are two parts to this dissertation. The first part deals mainly with multiclass classification, whereas the second part deals with regularized regression. In the following paragraphs a high-level overview of each of the chapters is given. For the uninitiated reader a brief introduction to two essential concepts, support vector machines and linear regression, will be given as well.

Part I: Multiclass Classification

In the chapters on multiclass classification, the support vector machine plays a central role. This technique, developed by Cortes and Vapnik (1995), finds the best separation line between data points from two classes. One way to gain an intuitive understanding of this algorithm is through the following analogy, in which we limit ourselves to cases where the data is perfectly separable. Consider two forests that lie close to each other. Each forest consists of one single type of tree, with one color of foliage (brown leaves and green leaves, for example). The support vector machine can be thought of as a way to find the broadest path that separates the two forests, such that all trees of one forest are on one side of the path and all trees of the other forest on the other side. Looking from the sky, one would see two groups of colored dots, separated by a path. The idea behind this method is that a wide path between the forests makes it easier to discern the types of trees compared to a narrow path. Extending the analogy further, the support vector machine allows for straight paths or wavy paths, but the goal remains to construct the broadest path possible.

In Chapter 2 the support vector machine algorithm is extended to deal with problems with two or more possible outcomes (in the analogy of the previous paragraph this means finding the best paths to separate two or more types of trees). The algorithm introduced in this chapter is a generalized multiclass support vector machine, called GenSVM. The motivation of this chapter comes from the observation that heuristic approaches to multiclass support vector machines are unsatisfactory due to their reliance on the binary SVM. Existing multiclass SVMs can require solving a large dual optimization problem and may not be sufficiently general due to specific choices made in the formulation of the loss function. GenSVM solves these issues by extending the binary SVM to multiclass problems while simultaneously generalizing several existing methods within a single formulation. The chapter further derives an optimization algorithm based on iterative majorization, which has the advantage of allowing for warm-starts during optimization of GenSVM for several hyperparameter settings. Finally, an extensive simulation study is performed that compares GenSVM with seven alternative multiclass SVMs. This study shows the performance of GenSVM in terms of predictive accuracy and illustrates the feasibility of the algorithm for large datasets. The paper on GenSVM was recently published in the Journal of Machine Learning Research (Van den Burg and Groenen, 2016).

Chapter 3 is concerned with estimating the Bayes error rate and its application to multiclass classification. In the support vector machine analogy above the Bayes error rate can be thought of as a measure of how difficult it is to find a path between the two forests. It is easier to find a path between forests that lie far apart than between forests that are intertwined. In recent work by Berisha et al. (2016) a nonparametric estimator of the Bayes error rate was presented that achieves considerably higher estimation accuracy than previous approaches. In Chapter 3 several practical improvements to this estimator are presented for the use in multiclass classification problems and meta-learning. Moreover, the estimator is applied to the multiclass classification problem as a way to construct a hierarchy of binary classification problems. In this hierarchy the "easier" hypotheses are decided upon first before progressing to the more difficult ones. The support vector machine is used as a binary classifier in this method, which leads to the formulation of the SmartSVM classification algorithm. This classifier is compared to several alternative multiclass SVMs in an experiment similar to that performed for the GenSVM classifier. This experiment shows the feasibility and usefulness of the SmartSVM classifier in practice.

Part II: Regularized Regression

The second part of this dissertation focuses on regularized linear regression algorithms. The linear regression method can be illustrated with the following hypothetical example. Consider a university that has done a survey among students to collect information about their studying behavior, such as their average grade, gender, age, number of study hours per week, hours of sleep per night, amount of coffee they drink, amount of alcohol consumed per week, and whether or not they are a member of a fraternity. The university may then be interested to see how these variables influence the average grade of a student. In this example the average grade of a student is the outcome variable and the other variables are the input variables. The assumption of linear regression is that a change in an input variable results in a proportional change in the outcome variable. For instance, a 10% increase in the number of study hours per week might give an increase of a certain percentage in the average grade, if all other factors remain the same. Alternatively, a 10% increase in the amount of alcohol consumed per week may result in a decrease of the average grade by a certain percentage. By performing a linear regression analysis the university can find a model for how each of the variables affects the average grade of the students. In this dissertation *sparse* linear regression is a significant topic. In sparse regression we are interested in the best model that explains the average grade with a limited number of variables. This is especially useful for situations where data is available for many variables, because it allows the university to find out which variables are the most important.

Finding the best regression model with a limited number of coefficients is a difficult problem in regularized regression. In Chapter 4 the SparseStep algorithm is presented to solve this problem based on the concept of graduated nonconvexity (Blake, 1983). This technique is based on the idea that local minima in the solution can be avoided if the nonconvexity in the problem is introduced slowly enough. In this chapter the SparseStep algorithm is introduced, an iterative majorization algorithm is derived, and an extensive simulation study is performed to investigate the performance of the algorithm in terms of modeling fidelity and predictive accuracy. The simulations show that SparseStep often outperforms the considered alternatives on these metrics.

In regularized regression it is common to limit the size of the estimated coefficients in some way. However, the way in which the size of the coefficients is measured has a significant influence on the obtained solution. For instance, the size can be measured as the sum of the absolute values of the coefficients or as the sum of squares of the coefficients. These are special cases of so-called ℓ_q penalties, with the sum of absolute values corresponding to an ℓ_1 penalty and the sum of squared values with an ℓ_2 penalty. Because of the properties of the obtained solution, ℓ_q-regularized regression is frequently used in practice. However, no single algorithm can solve the ℓ_q-regularized regression problem for all $q \in [0, 2]$ with the same formulation. In Chapter 5 the Smooth-q algorithm is presented that solves this exact problem by extending the SparseStep algorithm of Chapter 4 to all $q \in [0, 2]$. The main focus of this chapter is to establish the Smooth-q algorithm and to derive preliminary convergence results. The theoretical convergence results culminate in a currently unproven convergence conjecture, which states that parameters of the Smooth-q algorithm can always be chosen such that arbitrarily close convergence to the globally optimal solution is achieved. This conjecture is explored with numerical experiments for ℓ_0-regularized regression, which show that convergence to the global solution is achieved in a significant majority of the datasets.

1.2 Summary of Contributions

The main contributions of the individual chapters can be summarized as follows:

Chapter 2 introduces a generalized multiclass support vector machine which is called GenSVM (Van den Burg and Groenen, 2016).

Chapter 3 improves an accurate nonparametric estimator of the Bayes error rate and applies this estimator to meta-learning and hierarchical classifier design (Van den Burg and Hero, 2017).

Chapter 4 gives an iterative majorization algorithm for sparse regression, called SparseStep (Van den Burg et al., 2017).

Chapter 5 presents the Smooth-q algorithm for ℓ_q-regularized regression with $q \in [0,2]$.

In addition to the academic articles relating to these chapters, open-source software packages that implement the various methods are also contributions of this dissertation:

- For Chapter 2 a C library is available that implements the GenSVM method.[1]
- For Chapter 3 a Python package is released that implements the estimator and the hierarchical SmartSVM classifier.[2]
- For Chapter 4 an R library is available that implements the SparseStep algorithm.[3]
- Many experiments throughout this dissertation were performed on a compute cluster. The author has created a Python package to automate and simplify this process.[4]

A software package for the Smooth-q algorithm in Chapter 5 is planned, as well as implementations of some of the above methods in other programming languages.

1.3 Author Contributions

To conform to university regulations, the author contributions of the chapters are declared here. The dissertation author was responsible for writing the text of each of the chapters, with coauthors typically reviewing the writing and offering textual improvements where necessary. For Chapter 2 prof. Groenen offered additional suggestions for structuring the paper and positioning it in the existing literature. Chapter 3 was written in collaboration with prof. Hero. Chapters 4 and 5 were a collaboration with prof. Groenen and dr. Alfons. Research ideas were generally the product of an iterative process consisting of discussions with coauthors, empirical and mathematical experimentation, and evaluation of successful and unsuccessful research directions. All computer code necessary for the development and evaluation of each of the algorithms presented in this dissertation was written by the author.

[1]https://github.com/GjjvdBurg/GenSVM.
[2]https://github.com/HeroResearchGroup/SmartSVM.
[3]https://cran.r-project.org/web/packages/sparsestep/index.html.
[4]https://github.com/GjjvdBurg/abed.

I

Multiclass Classification

2

GenSVM: A Generalized Multiclass Support Vector Machine

G.J.J. van den Burg and P.J.F. Groenen

Abstract

Traditional extensions of the binary support vector machine (SVM) to multiclass problems are either heuristics or require solving a large dual optimization problem. Here, a generalized multiclass SVM is proposed called GenSVM. In this method classification boundaries for a K-class problem are constructed in a $(K-1)$-dimensional space using a simplex encoding. Additionally, several different weightings of the misclassification errors are incorporated in the loss function, such that it generalizes three existing multiclass SVMs through a single optimization problem. An iterative majorization algorithm is derived that solves the optimization problem without the need of a dual formulation. This algorithm has the advantage that it can use warm starts during cross validation and during a grid search, which significantly speeds up the training phase. Rigorous numerical experiments compare linear GenSVM with seven existing multiclass SVMs on both small and large datasets. These comparisons show that the proposed method is competitive with existing methods in both predictive accuracy and training time and that it significantly outperforms several existing methods on these criteria.

This chapter is based on Van den Burg and Groenen (2016).

2.1 Introduction

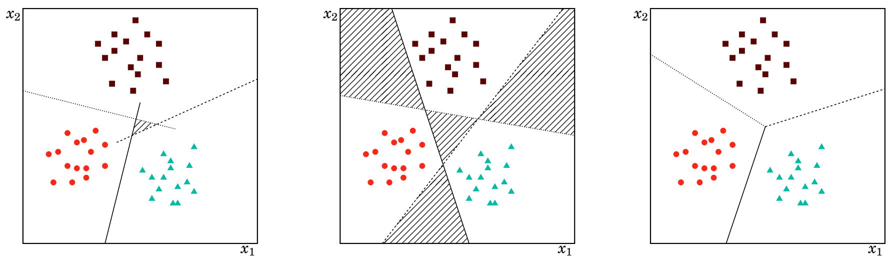
For binary classification, the support vector machine has shown to be very successful (Cortes and Vapnik, 1995). The SVM efficiently constructs linear or nonlinear classification boundaries and is able to yield a sparse solution through the so-called support vectors, that is, through those observations that are either not perfectly classified or are on the classification boundary. In addition, by regularizing the loss function the overfitting of the training dataset is curbed. Due to its desirable characteristics several attempts have been made to extend the SVM to classification problems where the number of classes K is larger than two. Overall, these extensions differ considerably in the approach taken to include multiple classes. Three types of approaches for multiclass SVMs (MSVMs) can be distinguished.

First, there are heuristic approaches that use the binary SVM as an underlying classifier and decompose the K-class problem into multiple binary problems. The most commonly used heuristic is the one-vs-one (OvO) method where decision boundaries are constructed between each pair of classes (Kreßel, 1999). OvO requires solving $K(K-1)$ binary SVM problems, which can be substantial if the number of classes is large. An advantage of OvO is that the problems to be solved are smaller in size. On the other hand, the one-vs-all (OvA) heuristic constructs K classification boundaries, one separating each class from all the other classes (Vapnik, 1998). Although OvA requires fewer binary SVMs to be estimated, the complete dataset is used for each classifier, which can create a high computational burden. Another heuristic approach is the directed acyclic graph (DAG) SVM proposed by Platt et al. (2000). DAGSVM is similar to the OvO approach except that the class prediction is done by successively voting away unlikely classes until only one remains. One problem with the OvO and OvA methods is that there are regions of the space for which class predictions are ambiguous, as illustrated in Figures 2.1(a) and 2.1(b).

In practice, heuristic methods such as the OvO and OvA approaches are used more often than other multiclass SVM implementations. One of the reasons for this is that there are several software packages that efficiently solve the binary SVM, such as LibSVM (Chang and Lin, 2011). This package implements a variation of the sequential minimal optimization algorithm of Platt (1999). Implementations of other multiclass SVMs in high-level (statistical) programming languages are lacking, which reduces their use in practice.[1]

The second type of extension of the binary SVM consists of methods that use error correcting codes. In these methods the problem is decomposed into

[1] An exception to this is the method of Lee et al. (2004), for which an R implementation exists. See http://www.stat.osu.edu/~yklee/software.html.



(a) One vs. One (b) One vs. All (c) Non-heuristic

FIGURE 2.1 – *Illustration of ambiguity regions for common heuristic multiclass SVMs. In the shaded regions* ties *occur for which no classification rule has been explicitly trained. Figure* (c) *corresponds to an SVM where all classes are considered simultaneously, which eliminates any possible ties. Figures inspired by Statnikov et al. (2011).*

multiple binary classification problems based on a constructed coding matrix that determines the grouping of the classes in a specific binary subproblem (Dietterich and Bakiri, 1995, Allwein et al., 2001, Crammer and Singer, 2002b). Error correcting code SVMs can thus be seen as a generalization of OvO and OvA. In Dietterich and Bakiri (1995) and Allwein et al. (2001), a coding matrix is constructed that determines which class instances are paired against each other for each binary SVM. Both approaches require that the coding matrix is determined beforehand. However, it is a priori unclear how such a coding matrix should be chosen. In fact, as Crammer and Singer (2002b) show, finding the optimal coding matrix is an NP-complete problem.

The third type of approaches are those that optimize one loss function to estimate all class boundaries simultaneously, the so-called single machine approaches (Rifkin and Klautau, 2004). In the literature, such methods have been proposed by, among others, Weston and Watkins (1998), Bredensteiner and Bennett (1999), Crammer and Singer (2002a), Lee et al. (2004), and Guermeur and Monfrini (2011). The method of Weston and Watkins (1998) yields a fairly large quadratic problem with a large number of slack variables, that is, $K-1$ slack variables for each observation.[2] The method of Crammer and Singer (2002a) reduces this number of slack variables by only penalizing the largest misclassification error. In addition, their method does not include a bias term in the decision boundaries, which is advantageous for solving the dual problem. Interestingly, this approach does not reduce parsimoniously to the binary SVM for $K = 2$. The method of Lee et al. (2004) uses a sum-to-zero constraint on the decision functions

[2]Slack variables are used in the optimization problem to capture inequality constraints. The number of slack variables is therefore a measure of the size of the optimization problem.

to reduce the dimensionality of the problem. This constraint effectively means that the solution of the multiclass SVM lies in a $(K-1)$-dimensional subspace of the full K dimensions considered. The size of the margins is reduced according to the number of classes, such that asymptotic convergence is obtained to the Bayes optimal decision boundary when the regularization term is ignored (Rifkin and Klautau, 2004). Finally, the method of Guermeur and Monfrini (2011) is a quadratic extension of the method developed by Lee et al. (2004). This extension keeps the sum-to-zero constraint on the decision functions, drops the nonnegativity constraint on the slack variables, and adds a quadratic function of the slack variables to the loss function. This means that at the optimum the slack variables are only positive on average, which differs from common SVM formulations.

The existing approaches to multiclass SVMs suffer from several problems. All current single machine multiclass extensions of the binary SVM rely on solving a potentially large dual optimization problem. This can be disadvantageous when a solution has to be found in a small amount of time, since iteratively improving the dual solution does not guarantee that the primal solution is improved as well. Thus, stopping early can lead to poor predictive performance. In addition, the dual of such single machine approaches should be solvable quickly in order to compete with existing heuristic approaches.

Almost all single machine approaches rely on misclassifications of the observed class with each of the other classes. By simply summing these misclassification errors (as in Lee et al., 2004) observations with multiple errors contribute more than those with a single misclassification do. Consequently, observations with multiple misclassifications have a stronger influence on the solution than those with a single misclassification, which is not a desirable property for a multiclass SVM, as it overemphasizes objects that are misclassified with respect to multiple classes. Here, it is argued that there is no reason to penalize certain misclassification regions more than others.

Single machine approaches are preferred for their ability to capture the multiclass classification problem in a single model. A parallel can be drawn here with multinomial regression and logistic regression. In this case, multinomial regression reduces exactly to the binary logistic regression method when $K = 2$, both techniques are single machine approaches, and many of the properties of logistic regression extend to multinomial regression. Therefore, it can be considered natural to use a single machine approach for the multiclass SVM that reduces parsimoniously to the binary SVM when $K = 2$.

The idea of casting the multiclass SVM problem to $K-1$ dimensions is appealing, since it reduces the dimensionality of the problem and is also present in other multiclass classification methods such as multinomial regression and linear discriminant analysis. However, the sum-to-zero constraint employed by Lee et al. (2004) creates an additional burden on the dual optimization problem (Dogan et al., 2011). Therefore, it would be desirable to cast the problem to $K-1$ dimensions in another manner. Below a simplex encoding will be introduced to achieve this goal. The simplex encoding for multiclass SVMs has been proposed earlier by Hill and Doucet (2007) and Mroueh et al. (2012), although the method outlined below differs from these two approaches. Note that the simplex coding approach by Mroueh et al. (2012) was shown to be equivalent to that of Lee et al. (2004) by Ávila Pires et al. (2013). An advantage of the simplex encoding is that in contrast to methods such as OvO and OvA, there are no regions of ambiguity in the prediction space (see Figure 2.1(c)). In addition, the low dimensional projection also has advantages for understanding the method, since it allows for a geometric interpretation. The geometric interpretation of existing single machine multiclass SVMs is often difficult since most are based on a dual optimization approach with little attention for a primal problem based on hinge errors.

A new flexible and general multiclass SVM is proposed, called GenSVM. This method uses the simplex encoding to formulate the multiclass SVM problem as a single optimization problem that reduces to the binary SVM when $K = 2$. By using a flexible hinge function and an ℓ_p norm of the errors the GenSVM loss function incorporates three existing multiclass SVMs that use the sum of the hinge errors, and extends these methods. In the linear version of GenSVM, $K-1$ linear combinations of the features are estimated next to the bias terms. In the nonlinear version, kernels can be used in a similar manner as can be done for binary SVMs. The resulting GenSVM loss function is convex in the parameters to be estimated. For this loss function an iterative majorization (IM) algorithm will be derived with guaranteed descent to the global minimum. By solving the optimization problem in the primal it is possible to use warm starts during a hyperparameter grid search or during cross validation, which makes the resulting algorithm very competitive in total training time, even for large datasets.

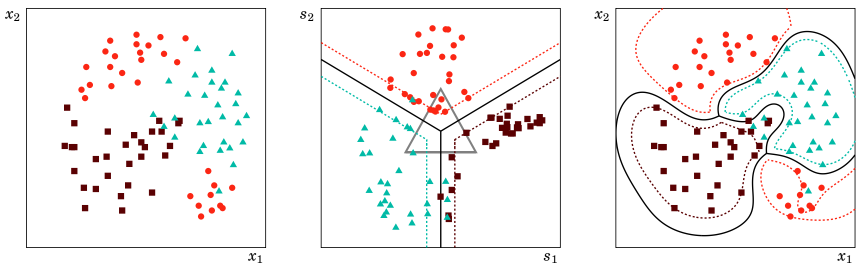
To evaluate its performance, GenSVM is compared to seven of the multiclass SVMs described above on several small datasets and one large dataset. The smaller datasets are used to assess the classification accuracy of GenSVM, whereas the large dataset is used to verify feasibility of GenSVM for large datasets. Due to the computational cost of these rigorous experiments only comparisons of linear multiclass SVMs are performed and experiments on nonlinear MSVMs are considered outside the scope of this chapter. Existing comparisons of multiclass SVMs in the literature do not determine any statistically significant differences in performance between classifiers and resort to tables of accuracy rates for the comparisons (for instance Hsu and Lin, 2002). Using suggestions from the benchmarking literature predictive performance and training time of all classifiers is compared using performance profiles and rank tests. The rank tests are used to uncover statistically significant differences between classifiers.

This chapter is organized as follows. Section 2.2 introduces the novel generalized multiclass SVM. In Section 2.3, features of the iterative majorization theory are reviewed and a number of useful properties are highlighted. Section 2.4 derives the IM algorithm for GenSVM, and presents pseudocode for the algorithm. Extensions of GenSVM to nonlinear classification boundaries are discussed in Section 2.5. A numerical comparison of GenSVM with existing multiclass SVMs on empirical datasets is done in Section 2.6. In Section 2.7 concluding remarks are provided.

2.2 The GenSVM Loss Function

Before introducing GenSVM formally, consider a small illustrative example of a hypothetical dataset of $n = 90$ objects with $K = 3$ classes and $m = 2$ attributes. Figure 2.2(a) shows the dataset in the space of these two attributes x_1 and x_2, with different classes denoted by different symbols. Figure 2.2(b) shows the $(K-1)$-dimensional simplex encoding of the data after an additional RBF kernel transformation has been applied and the mapping has been optimized to minimize misclassification errors (a detailed explanation follows). In this figure, the triangle shown in the center corresponds to a regular K-simplex in $K-1$ dimensions and the solid lines perpendicular to the faces of this simplex are the decision boundaries. This $(K-1)$-dimensional space will be referred to as the *simplex space* throughout this chapter. The mapping from the input space to this simplex space is optimized by minimizing the misclassification errors, which are calculated by measuring the distance of an object to the decision boundaries in the simplex space. Prediction of a class label is also done in this simplex space, by finding the nearest simplex vertex for the object. Figure 2.2(c) illustrates the decision boundaries in the original space of the input attributes x_1 and x_2. In Figures 2.2(b) and 2.2(c), the support vectors can be identified as the objects that lie on or beyond the dashed margin lines of their associated class. Note that the use of the simplex encoding ensures that for every point in the predictor space a class is predicted, hence no ambiguity regions can exist in the GenSVM solution.

The misclassification errors are formally defined as follows. Let $\mathbf{x}_i \in \mathbb{R}^m$ be an object vector corresponding to m attributes and let y_i denote the class label of object i with $y_i \in \{1,\ldots,K\}$, for $i \in \{1,\ldots,n\}$. Furthermore, let $\mathbf{W} \in \mathbb{R}^{m\times(K-1)}$ be a weight matrix and define a translation vector $\mathbf{t} \in \mathbb{R}^{K-1}$ for the bias terms. Then, object i is represented in the $(K-1)$-dimensional simplex space by $\mathbf{s}_i' = \mathbf{x}_i'\mathbf{W} + \mathbf{t}'$.



(a) Input space (b) Simplex space (c) Input space with boundaries

FIGURE 2.2 – *Illustration of GenSVM for a 2D dataset with $K = 3$ classes. In* (a) *the original data is shown, with different symbols denoting different classes. Figure* (b) *shows the mapping of the data to the $K-1$ dimensional simplex space, after an additional RBF kernel mapping has been applied and the optimal solution has been determined. The decision boundaries in this space are fixed as the perpendicular bisectors of the faces of the simplex, which is shown as the triangle. Figure* (c) *shows the resulting boundaries mapped back to the original input space, as can be seen by comparing with Figure* (a)*. In Figures* (b) *and* (c) *the dashed lines show the margins of the SVM solution.*

Note that here the *linear* version of GenSVM is described, the nonlinear version is described in Section 2.5.

To obtain the misclassification error of an object, the corresponding simplex space vector $\mathbf{s}'_i$ is projected on each of the decision boundaries that separate the true class of an object from another class. For the errors to be proportional with the distance to the decision boundaries, a regular K-simplex in $\mathbb{R}^{K-1}$ is used with distance 1 between each pair of vertices. Let $\mathbf{U}_K$ be the $K \times (K-1)$ coordinate matrix of this simplex, where a row $\mathbf{u}'_k$ of $\mathbf{U}_K$ gives the coordinates of a single vertex k. Then, it follows that with $k \in \{1,\dots,K\}$ and $l \in \{1,\dots,K-1\}$ the elements of $\mathbf{U}_K$ are given by

$$u_{kl} = \begin{cases} -\dfrac{1}{\sqrt{2(l^2+l)}} & \text{if } k \leq l \\ \dfrac{l}{\sqrt{2(l^2+l)}} & \text{if } k = l+1 \\ 0 & \text{if } k > l+1. \end{cases} \tag{2.1}$$

See Appendix 2.A for a derivation of this expression. Figure 2.3 shows an illustration of how the misclassification errors are computed for a single object. Consider object A with true class $y_A = 2$. It is clear that object A is misclassified as it is not located in the shaded area that has Vertex $\mathbf{u}_2$ as the nearest vertex.

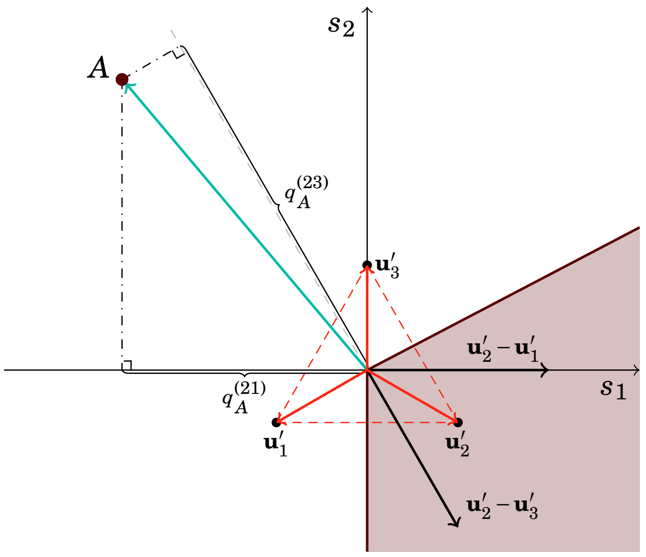


FIGURE 2.3 – *Graphical illustration of the calculation of distances $q_i^{(y_A j)}$ for an object A with $y_A = 2$ and $K = 3$. The figure shows the situation in the $(K-1)$-dimensional space. The distance $q_A^{(21)}$ is calculated by projecting $\mathbf{s}_A' = \mathbf{x}_A'\mathbf{W} + \mathbf{t}'$ on $\mathbf{u}_2 - \mathbf{u}_1$ and the distance $q_A^{(23)}$ is found by projecting $\mathbf{s}_A'$ on $\mathbf{u}_2 - \mathbf{u}_3$. The boundary between the class 1 and class 3 regions has been omitted for clarity, but lies along $\mathbf{u}_2$.*

The boundaries of the shaded area are given by the perpendicular bisectors of the edges of the simplex between Vertices $\mathbf{u}_2$ and $\mathbf{u}_1$ and between Vertices $\mathbf{u}_2$ and $\mathbf{u}_3$, and form the decision boundaries for class 2. The error for object A is computed by determining the distance from the object to each of these decision boundaries. Let $q_A^{(21)}$ and $q_A^{(23)}$ denote these distances to the class boundaries, which are obtained by projecting $\mathbf{s}_A' = \mathbf{x}_A'\mathbf{W} + \mathbf{t}'$ on $\mathbf{u}_2 - \mathbf{u}_1$ and $\mathbf{u}_2 - \mathbf{u}_3$ respectively, as illustrated in the figure. Generalizing this reasoning, scalars $q_i^{(kj)}$ can be defined to measure the projection distance of object i onto the boundary between class k and j in the simplex space, as

$$q_i^{(kj)} = (\mathbf{x}_i'\mathbf{W} + \mathbf{t}')(\mathbf{u}_k - \mathbf{u}_j). \tag{2.2}$$

It is required that the GenSVM loss function is both general and flexible, such that it can easily be tuned for the specific dataset at hand. To achieve this, a loss function is constructed with a number of different weightings, each with a specific effect on the object distances $q_i^{(kj)}$. In the proposed loss function flexibility is added through the use of the Huber hinge function instead of the absolute hinge function and by using the ℓ_p norm of the hinge errors instead of the sum. The motivation for these choices follows.

As is customary for SVMs a hinge loss is used to ensure that instances that do not cross their class margin will yield zero error. Here, the flexible and continuous Huber hinge loss is used (after the Huber error in robust statistics, see Huber, 1964), which is defined as

$$h(q) = \begin{cases} 1 - q - \frac{\kappa + 1}{2} & \text{if } q \leq -\kappa \\ \frac{1}{2(\kappa+1)}(1-q)^2 & \text{if } q \in (-\kappa, 1] \\ 0 & \text{if } q > 1, \end{cases} \tag{2.3}$$

with $\kappa > -1$. The Huber hinge loss has been independently introduced in Chapelle (2007), Rosset and Zhu (2007), and Groenen et al. (2008). This hinge error is zero when an instance is classified correctly with respect to its class margin. However, in contrast to the absolute hinge error, it is continuous due to a quadratic region in the interval $(-\kappa, 1]$. This quadratic region allows for a softer weighting of objects close to the decision boundary. Additionally, the smoothness of the Huber hinge error is a desirable property for the iterative majorization algorithm derived in Section 2.4.1. Note that the Huber hinge error approaches the absolute hinge for $\kappa \downarrow -1$ and the quadratic hinge for $\kappa \to \infty$.

The Huber hinge error is applied to each of the distances $q_i^{(y_i j)}$, for $j \neq y_i$. Thus, no error is counted when the object is correctly classified. For each of the objects, errors with respect to the other classes are summed using an ℓ_p norm to obtain the total object error

$$\left(\sum_{\substack{j=1 \\ j \neq y_i}}^{K} h^p \left(q_i^{(y_i j)} \right) \right)^{\frac{1}{p}}. \tag{2.4}$$

The ℓ_p norm is added to provide a form of regularization on Huber weighted errors for instances that are misclassified with respect to multiple classes. As argued in the Introduction, simply summing misclassification errors can lead to overemphasizing of instances with multiple misclassification errors. By adding an ℓ_p norm of the hinge errors the influence of such instances on the loss function can be tuned. With the addition of the ℓ_p norm on the hinge errors it is possible to illustrate how GenSVM generalizes existing methods. For instance, with $p = 1$ and $\kappa \downarrow -1$, the loss function solves the same problem as Lee et al. (2004). Next, for $p = 2$ and $\kappa \downarrow -1$ it resembles that of Guermeur and Monfrini (2011). Finally, for $p = \infty$ and $\kappa \downarrow -1$ the ℓ_p norm reduces to the max norm of the hinge errors, which corresponds to the method of Crammer and Singer (2002a). Note that in each case the value of κ can additionally be varied to include an even broader family of loss functions.

To illustrate the effects of p and κ on the total object error, refer to Figure 2.4. In Figures 2.4(a) and 2.4(b), the value of p is set to $p = 1$ and $p = 2$ respectively, while maintaining the absolute hinge error using $\kappa = -0.95$. A reference point is plotted at a fixed position in the area of the simplex space where there is a nonzero error with respect to two classes. It can be seen from this reference point that the value of the combined error is higher when $p = 1$. With $p = 2$ the combined error at the reference point approximates the Euclidean distance to the margin, when $\kappa \downarrow -1$. Figures 2.4(a), 2.4(c), and 2.4(d) show the effect of varying κ. It can be seen that the error near the margin becomes more quadratic with increasing κ. In fact, as κ increases the error approaches the squared Euclidean distance to the margin, which can be used to obtain a quadratic hinge multiclass SVM. Both of these effects will become stronger when the number of classes increases, as increasingly more objects will have errors with respect to more than one class.

Next, let $\rho_i \geq 0$ denote optional object weights, which are introduced to allow flexibility in the way individual objects contribute to the total loss function. With these individual weights it is possible to correct for different group sizes, or to give additional weights to misclassifications of certain classes. When correcting for group sizes, the weights can be chosen as

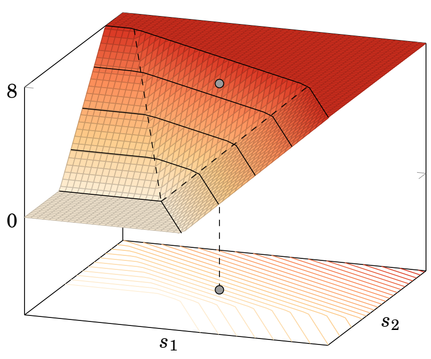
$$\rho_i = \frac{n}{n_k K}, \quad i \in G_k, \tag{2.5}$$

where $G_k = \{i : y_i = k\}$ is the set of objects belonging to class k, and $n_k = |G_k|$. The complete GenSVM loss function combining all n objects can now be formulated as

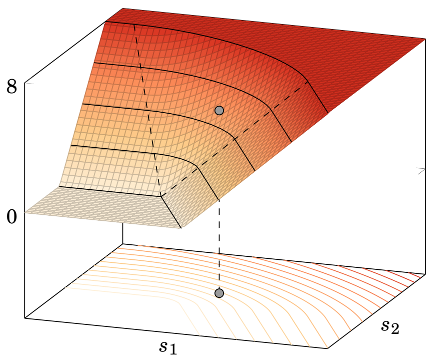
$$L_{\text{MSVM}}(\mathbf{W}, \mathbf{t}) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in G_k} \rho_i \left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right)^{\frac{1}{p}} + \lambda \operatorname{tr} \mathbf{W}'\mathbf{W}, \tag{2.6}$$

where $\lambda \operatorname{tr} \mathbf{W}'\mathbf{W}$ is the penalty term to avoid overfitting and $\lambda > 0$ is the regularization parameter. Note that for the case where $K = 2$, the above loss function reduces to the loss function for binary SVM given in Groenen et al. (2008), with Huber hinge errors.

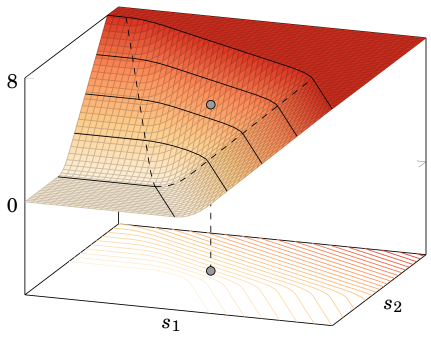
The outline of a proof for the convexity of the loss function in (2.6) is given. First, note that the distances $q_i^{(kj)}$ in the loss function are affine in $\mathbf{W}$ and $\mathbf{t}$. Hence, if the loss function is convex in $q_i^{(kj)}$ it is convex in $\mathbf{W}$ and $\mathbf{t}$ as well. Second, the Huber hinge function is trivially convex in $q_i^{(kj)}$, since each separate



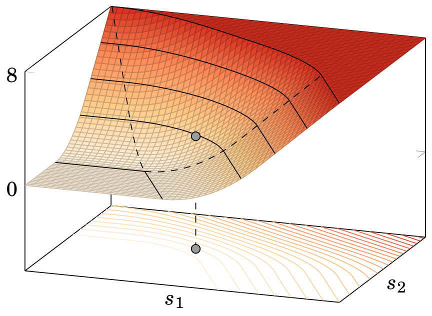
(a) $p = 1$ and $\kappa = -0.95$



(b) $p = 2$ and $\kappa = -0.95$



(c) $p = 1$ and $\kappa = 1.0$



(d) $p = 1$ and $\kappa = 5.0$

FIGURE 2.4 – *Illustration of the ℓ_p norm of the Huber weighted errors. Comparing figures* (a) *and* (b) *shows the effect of the ℓ_p norm. With $p = 1$ objects that have errors w.r.t. both classes are penalized more strongly than those with only one error, whereas with $p = 2$ this is not the case. Figures* (a), (c), *and* (d) *compare the effect of the κ parameter, with $p = 1$. This shows that with a large value of κ, the errors close to the boundary are weighted quadratically. Note that s_1 and s_2 indicate the dimensions of the simplex space.*

piece of the function is convex and the Huber hinge is continuous. Third, the ℓ_p norm is a convex function by the Minkowski inequality and it is monotonically increasing by definition. Thus, it follows that the ℓ_p norm of the Huber weighted instance errors is convex (see for instance Rockafellar, 1997). Next, since it is required that the weights ρ_i are non-negative, the sum in the first term of (2.6) is a convex combination. Finally, the penalty term can also be shown to be convex, since $\text{tr}\,\mathbf{W}'\mathbf{W}$ is the square of the Frobenius norm of $\mathbf{W}$ and it is required that $\lambda > 0$. Thus, it holds that the loss function in (2.6) is convex in $\mathbf{W}$ and $\mathbf{t}$.

Predicting class labels in GenSVM can be done as follows. Let $(\mathbf{W}^*, \mathbf{t}^*)$ denote the parameters that minimize the loss function. Predicting the class label of an unseen sample $\mathbf{x}'_{n+1}$ can then be done by first mapping it to the simplex space, using the optimal projection: $\mathbf{s}'_{n+1} = \mathbf{x}'_{n+1}\mathbf{W}^* + \mathbf{t}'^*$. The predicted class label is then simply the label corresponding to the nearest simplex vertex as measured by the squared Euclidean norm, or

$$\hat{y}_{n+1} = \arg\min_k \|\mathbf{s}'_{n+1} - \mathbf{u}'_k\|^2, \quad \text{for } k = 1, \ldots, K. \tag{2.7}$$

2.3 Iterative Majorization

To minimize the loss function given in (2.6), an iterative majorization (IM) algorithm will be derived. Iterative majorization was first described by Weiszfeld (1937), however the first application of the algorithm in the context of a line search comes from Ortega and Rheinboldt (1970, p. 253—255). During the late 1970s, the method was independently developed by De Leeuw (1977) as part of the SMACOF algorithm for multidimensional scaling and by Voss and Eckhardt (1980) as a general minimization method. For the reader unfamiliar with the iterative majorization algorithm a more detailed description has been included in Appendix 2.B and further examples can be found in for instance Hunter and Lange (2004).

The asymptotic convergence rate of the IM algorithm is linear, which is less than that of the Newton-Raphson algorithm (De Leeuw, 1994). However, the largest improvements in the loss function will occur in the first few steps of the iterative majorization algorithm, where the asymptotic linear rate does not apply (Havel, 1991). This property will become very useful for GenSVM as it allows for a quick approximation to the exact SVM solution in few iterations.

There is no straightforward technique for deriving the majorization function for any given function. However, in the next section the derivation of the majorization function for the GenSVM loss function is presented using an "outside-in" approach. In this approach, each function that constitutes the loss function is majorized separately and the majorization functions are combined. Two properties of majorization functions that are useful for this derivation are now formally defined. In these expressions, $\overline{x}$ is a supporting point, as defined in Appendix 2.B.

P1. Let $f_1 : \mathcal{Y} \to \mathcal{Z}$, $f_2 : \mathcal{X} \to \mathcal{Y}$ and define $f = f_1 \circ f_2 : \mathcal{X} \to \mathcal{Z}$, such that for $x \in \mathcal{X}$, $f(x) = f_1(f_2(x))$. If $g_1 : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Z}$ is a majorization function of f_1, then $g : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$ defined as $g = g_1 \circ f_2$ is a majorization function of f. Thus for $x, \overline{x} \in \mathcal{X}$ it holds that $g(x, \overline{x}) = g_1(f_2(x), f_2(\overline{x}))$ is a majorization function of $f(x)$ at $\overline{x}$.

P2. Let $f_i : \mathcal{X} \to \mathcal{Z}$ and define $f : \mathcal{X} \to \mathcal{Z}$ such that $f(x) = \sum_i a_i f_i(x)$ for $x \in \mathcal{X}$, with $a_i \geq 0$ for all i. If $g_i : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$ is a majorization function for f_i at a point $\overline{x} \in \mathcal{X}$, then $g : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$ given by $g(x, \overline{x}) = \sum_i a_i g_i(x, \overline{x})$ is a majorization function of f.

Proofs of these properties are omitted, as they follow directly from the requirements for a majorization function given in Appendix 2.B.

2.4 GenSVM Optimization and Implementation

In this section, a quadratic majorization function for GenSVM will be derived. Although it is possible to derive a majorization algorithm for general values of the ℓ_p norm parameter, the following derivation will restrict this value to the interval $p \in [1, 2]$ because this avoids the issue that quadratic majorization can become slow for $p > 2$, and because it simplifies the derivation.[3] Pseudocode for the derived algorithm will be presented, as well as an analysis of the computational complexity of the algorithm. Finally, an important remark on the use of warm starts in the algorithm is given.

2.4.1 *Majorization Derivation*

To shorten the notation, let $\mathbf{V} = [\mathbf{t}\ \mathbf{W}']'$, $\mathbf{z}_i' = [1\ \mathbf{x}_i']$, and $\boldsymbol{\delta}_{kj} = \mathbf{u}_k - \mathbf{u}_j$, such that $q_i^{(kj)} = \mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}$. With this notation it becomes sufficient to optimize the loss function with respect to $\mathbf{V}$. Formulated in this manner (2.6) becomes

$$L_{\text{MSVM}}(\mathbf{V}) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in G_k} \rho_i \left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right)^{\frac{1}{p}} + \lambda \operatorname{tr} \mathbf{V}'\mathbf{J}\mathbf{V}, \tag{2.8}$$

where $\mathbf{J}$ is an $m+1$ diagonal matrix with $\mathbf{J}_{i,i} = 1$ for $i > 1$ and zero elsewhere. To derive a majorization function for this expression the "outside-in" approach will be used, together with the properties of majorization functions. In what follows, variables with a bar denote supporting points for the IM algorithm. The goal of the derivation is to find a quadratic majorization function in $\mathbf{V}$ such that

$$L_{\text{MSVM}}(\mathbf{V}) \leq \operatorname{tr} \mathbf{V}'\mathbf{Z}'\mathbf{A}\mathbf{Z}'\mathbf{V} - 2 \operatorname{tr} \mathbf{V}'\mathbf{Z}'\mathbf{B} + C, \tag{2.9}$$

where $\mathbf{A}$, $\mathbf{B}$, and C are coefficients of the majorization depending on $\overline{\mathbf{V}}$. The matrix $\mathbf{Z}$ is simply the $n \times (m+1)$ matrix with rows $\mathbf{z}_i'$.

Property P2 above means that the summation over instances in the loss function can be ignored for now. Moreover, since the regularization term is

[3]For a majorization algorithm of the ℓ_p norm with $p \geq 2$, see Groenen et al. (1999).

quadratic in $\mathbf{V}$ it requires no majorization. The outermost function for which a majorization function has to be found is thus the ℓ_p norm of the Huber hinge errors. A majorization function for the ℓ_p norm could be constructed, but a discontinuity in the derivative will exist at the origin (Tsutsu and Morikawa, 2012). To avoid this discontinuity in the derivative of the ℓ_p norm, the following inequality is needed (Hardy et al., 1934, eq. 2.10.3)

$$\left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right)^{\frac{1}{p}} \leq \sum_{j \neq k} h \left(q_i^{(kj)} \right). \tag{2.10}$$

This inequality can be used as a majorization function only if equality holds at the supporting point,

$$\left(\sum_{j \neq k} h^p \left(\overline{q}_i^{(kj)} \right) \right)^{\frac{1}{p}} = \sum_{j \neq k} h \left(\overline{q}_i^{(kj)} \right). \tag{2.11}$$

It is straightforward to see that this only holds if at most one of the $h\left(\overline{q}_i^{(kj)}\right)$ errors is nonzero for $j \neq k$. Thus an indicator variable ε_i is introduced which is 1 if at most one of these errors is nonzero and 0 otherwise. Then it follows that

$$\begin{aligned} L_{\text{MSVM}}(\mathbf{V}) \leq & \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in G_k} \rho_i \left[\varepsilon_i \sum_{j \neq k} h \left(q_i^{(kj)} \right) + (1 - \varepsilon_i) \left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right)^{\frac{1}{p}} \right] \\ & + \lambda \operatorname{tr} \mathbf{V}'\mathbf{J}\mathbf{V}. \end{aligned} \tag{2.12}$$

Now, the next function for which a majorization needs to be found is $f_1(x) = x^{1/p}$. From the inequality $a^\alpha b^\beta < \alpha a + \beta b$, with $\alpha + \beta = 1$ (Hardy et al., 1934, Theorem 37), a linear majorization inequality can be constructed for this function by substituting $a = x$, $b = \overline{x}$, $\alpha = \frac{1}{p}$ and $\beta = 1 - \frac{1}{p}$ (Groenen and Heiser, 1996). This yields

$$f_1(x) = x^{\frac{1}{p}} \leq \frac{1}{p} \overline{x}^{\frac{1}{p} - 1} x + \left(1 - \frac{1}{p} \right) \overline{x}^{\frac{1}{p}} = g_1(x, \overline{x}). \tag{2.13}$$

Applying this majorization and using property P1 gives

$$\left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right)^{\frac{1}{p}} \leq \frac{1}{p} \left(\sum_{j \neq k} h^p \left(\overline{q}_i^{(kj)} \right) \right)^{\frac{1}{p} - 1} \left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right) + \left(1 - \frac{1}{p} \right) \left(\sum_{j \neq k} h^p \left(\overline{q}_i^{(kj)} \right) \right)^{\frac{1}{p}}.$$

Plugging this into (2.12) and collecting terms yields,

$$L_{\text{MSVM}}(\mathbf{V}) \leq \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in G_k} \rho_i \left[\varepsilon_i \sum_{j \neq k} h \left(q_i^{(kj)} \right) + (1 - \varepsilon_i) \omega_i \sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right] + \Gamma^{(1)} + \lambda \operatorname{tr} \mathbf{V}'\mathbf{J}\mathbf{V}$$

with

$$\omega_i = \frac{1}{p}\left(\sum_{j\neq k} h^p\left(\overline{q}_i^{(kj)}\right)\right)^{\frac{1}{p}-1}. \tag{2.14}$$

The constant $\Gamma^{(1)}$ contains all terms that only depend on previous errors $\overline{q}_i^{(kj)}$.

The next majorization step by the "outside-in" approach is to find a quadratic majorization function for $f_2(x) = h^p(x)$, of the form

$$f_2(x) = h^p(x) \leq a(\overline{x}, p)x^2 - 2b(\overline{x}, p)x + c(\overline{x}, p) = g_2(x, \overline{x}). \tag{2.15}$$

For brevity this derivation has been moved to Appendix 2.C. In the remainder of this derivation, $a_{ijk}^{(p)}$ will be used to abbreviate $a\left(\overline{q}_i^{(kj)}, p\right)$, with similar abbreviations for b and c. Using these majorizations and making the dependence on $\mathbf{V}$ explicit by substituting $q_i^{(kj)} = \mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}$ gives

$$\begin{aligned} L_{\text{MSVM}}(\mathbf{V}) \leq{}& \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k} \rho_i\varepsilon_i \sum_{j\neq k}\left[a_{ijk}^{(1)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'\mathbf{V}'\mathbf{z}_i - 2b_{ijk}^{(1)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\right] \\ &+ \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k} \rho_i(1-\varepsilon_i)\omega_i \sum_{j\neq k}\left[a_{ijk}^{(p)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'\mathbf{V}'\mathbf{z}_i - 2b_{ijk}^{(p)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\right] + \Gamma^{(2)} + \lambda \operatorname{tr} \mathbf{V}'\mathbf{J}\mathbf{V}, \end{aligned} \tag{2.16}$$

where $\Gamma^{(2)}$ again contains all constant terms. Due to dependence on the matrix $\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'$, the above majorization function is not yet in the desired quadratic form of (2.9). However, since the maximum eigenvalue of $\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'$ is 1 by definition of the simplex coordinates, it follows that the matrix $\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}' - \mathbf{I}$ is negative semidefinite. Hence, it can be shown that the inequality $\mathbf{z}_i'(\mathbf{V}-\overline{\mathbf{V}})(\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}' - \mathbf{I})(\mathbf{V}-\overline{\mathbf{V}})'\mathbf{z}_i \leq 0$ holds (Bijleveld and De Leeuw, 1991, Theorem 4). Rewriting this gives the majorization inequality

$$\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'\mathbf{V}'\mathbf{z}_i \leq \mathbf{z}_i'\mathbf{V}\mathbf{V}'\mathbf{z}_i - 2\mathbf{z}_i'\mathbf{V}(\mathbf{I} - \boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}')\overline{\mathbf{V}}\mathbf{z}_i + \mathbf{z}_i'\overline{\mathbf{V}}(\mathbf{I} - \boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}')\overline{\mathbf{V}}'\mathbf{z}_i. \tag{2.17}$$

With this inequality the majorization inequality becomes

$$\begin{aligned} L_{\text{MSVM}}(\mathbf{V}) \leq{}& \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k} \rho_i \mathbf{z}_i'\mathbf{V}(\mathbf{V}' - 2\overline{\mathbf{V}}')\mathbf{z}_i \sum_{j\neq k}\left[\varepsilon_i a_{ijk}^{(1)} + (1-\varepsilon_i)\omega_i a_{ijk}^{(p)}\right] \\ &- \frac{2}{n}\sum_{k=1}^{K}\sum_{i\in G_k} \rho_i \mathbf{z}_i'\mathbf{V}\sum_{j\neq k}\left[\varepsilon_i\left(b_{ijk}^{(1)} - a_{ijk}^{(1)}\overline{q}_i^{(kj)}\right) + (1-\varepsilon_i)\omega_i\left(b_{ijk}^{(p)} - a_{ijk}^{(p)}\overline{q}_i^{(kj)}\right)\right]\boldsymbol{\delta}_{kj} \\ &+ \Gamma^{(3)} + \lambda \operatorname{tr} \mathbf{V}'\mathbf{J}\mathbf{V}, \end{aligned} \tag{2.18}$$

where $\overline{q}_i^{(kj)} = \mathbf{z}_i'\overline{\mathbf{V}}\boldsymbol{\delta}_{kj}$. This majorization function is quadratic in $\mathbf{V}$ and can thus be used in the IM algorithm. To derive the first-order condition used in the update step of the IM algorithm (step 2 in Appendix 2.B), matrix notation for the above expression is introduced. Let $\mathbf{A}$ be an $n \times n$ diagonal matrix with elements α_i and let $\mathbf{B}$ be an $n \times (K-1)$ matrix with rows $\boldsymbol{\beta}'_i$, where

$$\alpha_i = \frac{1}{n}\rho_i \sum_{j \neq k} \left[\varepsilon_i a^{(1)}_{ijk} + (1-\varepsilon_i)\omega_i a^{(p)}_{ijk} \right], \tag{2.19}$$

$$\boldsymbol{\beta}'_i = \frac{1}{n}\rho_i \sum_{j \neq k} \left[\varepsilon_i \left(b^{(1)}_{ijk} - a^{(1)}_{ijk} \overline{q}^{(kj)}_i \right) + (1-\varepsilon_i)\omega_i \left(b^{(p)}_{ijk} - a^{(p)}_{ijk} \overline{q}^{(kj)}_i \right) \right] \boldsymbol{\delta}'_{kj}. \tag{2.20}$$

Then the majorization function of $L_{\text{MSVM}}(\mathbf{V})$ given in (2.18) can be written as

$$L_{\text{MSVM}}(\mathbf{V}) \leq \text{tr}\,(\mathbf{V} - 2\overline{\mathbf{V}})'\mathbf{Z}'\mathbf{AZV} - 2\,\text{tr}\,\mathbf{B}'\mathbf{ZV} + \Gamma^{(3)} + \lambda\,\text{tr}\,\mathbf{V}'\mathbf{JV} \tag{2.21}$$

$$= \text{tr}\,\mathbf{V}'(\mathbf{Z}'\mathbf{AZ} + \lambda\mathbf{J})\mathbf{V} - 2\,\text{tr}\,(\overline{\mathbf{V}}'\mathbf{Z}'\mathbf{A} + \mathbf{B}')\mathbf{ZV} + \Gamma^{(3)}. \tag{2.22}$$

This majorization function has the desired functional form described in (2.9). Differentiation with respect to $\mathbf{V}$ and equating to zero yields the linear system

$$(\mathbf{Z}'\mathbf{AZ} + \lambda\mathbf{J})\mathbf{V} = \mathbf{Z}'\mathbf{AZ}\overline{\mathbf{V}} + \mathbf{Z}'\mathbf{B}. \tag{2.23}$$

The update $\mathbf{V}^+$ that solves this system can then be calculated efficiently by Gaussian elimination.

2.4.2 *Algorithm Implementation and Complexity*

Pseudocode for the main GenSVM function is given in Algorithm 2.1. As can be seen, the algorithm simply updates all instance coefficients at each iteration using the function presented in Algorithm 2.2, which computes the updates for α_i and $\boldsymbol{\beta}_i$. In practice, some calculations can be done more efficiently for all instances by using matrix algebra. When step doubling (see Appendix 2.B) is applied in the majorization algorithm, line 17 is replaced by $\mathbf{V} \leftarrow 2\mathbf{V}^+ - \overline{\mathbf{V}}$. In the implementation step doubling is applied after a burn-in of 50 iterations. The implementation used in the experiments described in Section 2.6 is written in `C`, using the BLAS and LAPACK libraries. The source code for this `C` library is available under the open source GNU GPL license, through an online repository. A thorough description of the implementation is available in the package documentation.

The complexity of a single iteration of the IM algorithm is $O(n(m+1)^2)$ assuming that $n > m > K$. As noted earlier, the convergence rate of the general IM algorithm is linear. Computational complexity of standard SVM solvers that solve the dual problem through decomposition methods lies between $O(n^2)$ and $O(n^3)$ depending on the value of λ (Bottou and Lin, 2007). An efficient algorithm

Algorithm 2.1. GenSVM

```
1: function GenSVM($\mathbf{X}, \mathbf{y}, \boldsymbol{\rho}, p, \kappa, \lambda, \epsilon$)
2:     $K \leftarrow \max\{\mathbf{y}\}$
3:     $t \leftarrow 1$
4:     $\mathbf{Z} \leftarrow [\mathbf{1}\ \mathbf{X}]$
5:     $\overline{\mathbf{V}} \leftarrow \mathbf{V}_0$                    ▷ $\mathbf{V}_0$ is randomly generated or a warm-start
6:     Generate $\mathbf{J}$ and $\mathbf{U}_K$
7:     $L_t \leftarrow L_{\text{MSVM}}(\overline{\mathbf{V}})$
8:     $L_{t-1} \leftarrow (1+2\epsilon)L_t$
9:     while $(L_{t-1} - L_t)/L_t > \epsilon$ do
10:        for $i \leftarrow 1, n$ do
11:            $\alpha_i, \beta_i \leftarrow$ GenSVMCoef($\mathbf{z}_i, y_i, \rho_i, p, \kappa, \mathbf{U}_K$)
12:        end for
13:        Construct $\mathbf{A}$ from $\alpha_i$
14:        Construct $\mathbf{B}$ from $\boldsymbol{\beta}_i$
15:        Find $\mathbf{V}^+$ that solves (2.23)
16:        $\overline{\mathbf{V}} \leftarrow \mathbf{V}$
17:        $\mathbf{V} \leftarrow \mathbf{V}^+$
18:        $L_{t-1} \leftarrow L_t$
19:        $L_t \leftarrow L_{\text{MSVM}}(\mathbf{V})$
20:        $t \leftarrow t+1$
21:    end while
22:    return $\mathbf{V}$
23: end function
```

for the method of Crammer and Singer (2002a) developed by Keerthi et al. (2008) has a complexity of $O(n\overline{m}K)$ per iteration, where $\overline{m} \leq m$ is the average number of nonzero features per training instance. In the methods of Lee et al. (2004) and Weston and Watkins (1998), a quadratic programming problem with $n(K-1)$ dual variables needs to be solved, which is typically done using a standard solver. An analysis of the exact convergence of GenSVM, including the expected number of iterations needed to achieve convergence at a factor ϵ, is outside the scope of the current work and a subject for further research.

2.4.3 *Smart Initialization*

When training machine learning algorithms to determine the optimal hyperparameters, it is common to use cross validation (CV). With GenSVM it is possible to initialize the matrix $\overline{\mathbf{V}}$ such that the final result of a fold is used as the initial value for $\mathbf{V}_0$ for the next fold. This same technique can be used when searching for the optimal hyperparameter configuration in a grid search, by initializing the weight matrix with the outcome of the previous configuration. Such warm-start

Algorithm 2.2. GenSVM Instance Coefficients

```
1: function GENSVMCOEF($\mathbf{z}_i, y_i, \rho_i, p, \kappa, \mathbf{U}_K$)
2:     Compute $\overline{q}_i^{(y_i j)} = \mathbf{z}_i'\overline{\mathbf{V}}\boldsymbol{\delta}_{y_i j}$ for all $j \neq y_i$
3:     Compute $h\left(\overline{q}_i^{(y_i j)}\right)$ for all $j \neq y_i$ by (2.3)
4:     Determine $\varepsilon_i$
5:     if $\varepsilon_i = 1$ then
6:         Compute $a_{ijy_i}^{(1)}$ and $b_{ijy_i}^{(1)}$ for $j \neq y_i$ according to Table 2.4
7:     else
8:         Compute $\omega_i$ following (2.14)
9:         Compute $a_{ijy_i}^{(p)}$ and $b_{ijy_i}^{(p)}$ for $j \neq y_i$ according to Table 2.4
10:    end if
11:    Compute $\alpha_i$ by (2.19)
12:    Compute $\boldsymbol{\beta}_i$ by (2.20)
13:    return $\alpha_i$, $\boldsymbol{\beta}_i$
14: end function
```

initialization greatly reduces the time needed to perform cross validation with GenSVM. It is important to note here that using warm starts is not easily possible with dual optimization approaches. Therefore, the ability to use warm starts can be seen as an advantage of solving the GenSVM optimization problem in the primal.

2.5 NONLINEARITY

One possible method to include nonlinearity in a classifier is through the use of spline transformations (see for instance Hastie et al., 2009). With spline transformations each attribute vector $\mathbf{x}_j$ is transformed to a spline basis $\mathbf{N}_j$, for $j = 1, \ldots, m$. The transformed input matrix $\mathbf{N} = [\mathbf{N}_1, \ldots, \mathbf{N}_m]$ is then of size $n \times l$, where l depends on the degree of the spline transformation and the chosen number of interior knots of the spline. An application of spline transformations to the binary SVM can be found in Groenen et al. (2007).

A more common way to include nonlinearity in machine learning methods is through the use of the kernel trick, attributed to Aizerman et al. (1964). With the kernel trick, the dot product of two instance vectors in the dual optimization problem is replaced by the dot product of the same vectors in a high dimensional feature space. Since no dot products appear in the primal formulation of GenSVM, a different method is used here. By applying a preprocessing step on the kernel matrix, nonlinearity can be included using the same algorithm as the one presented for the linear case. Furthermore, predicting class labels requires a postprocessing step on the obtained matrix $\mathbf{V}^*$. A full derivation is given in Appendix 2.D.

2.6 Experiments

To assess the performance of the proposed GenSVM classifier, a simulation study was done comparing GenSVM with seven existing multiclass SVMs on 13 small datasets. These experiments are used to precisely measure predictive accuracy and total training time using performance profiles and rank plots. To verify the feasibility of GenSVM for large datasets an additional simulation study is done. The results of this study are presented separately in Section 2.6.4. Due to the large number of datasets and methods involved, experiments were only done for the linear kernel. Experiments on nonlinear multiclass SVMs would require even more training time than for linear MSVMs and is considered outside the scope of this chapter.

2.6.1 *Setup*

Implementations of the heuristic multiclass SVMs (OvO, OvA, and DAG) were included through LibSVM (v. 3.16, Chang and Lin, 2011). LibSVM is a popular library for binary SVMs with packages for many programming languages, it is written in `C++` and implements a variation of the SMO algorithm of Platt (1999). The OvO and DAG methods are implemented in this package, and a `C` implementation of OvA using LibSVM was created for these experiments.[4] For the single-machine approaches the MSVMpack package was used (v. 1.3, Lauer and Guermeur, 2011), which is written in `C`. This package implements the methods of Weston and Watkins (W&W, 1998), Crammer and Singer (C&S, 2002a), Lee et al. (LLW, 2004), and Guermeur and Monfrini (MSVM^2, 2011). Finally, to verify if implementation differences are relevant for algorithm performance the LibLinear (Fan et al., 2008) implementation of the method by Crammer and Singer (2002a) is also included (denoted LL C&S). This implementation uses the optimization algorithm by Keerthi et al. (2008).

To compare the classification methods properly, it is desirable to remove any bias that could occur when using cross validation (Cawley and Talbot, 2010). Therefore, *nested* cross validation is used (Stone, 1974), as illustrated in Figure 2.5. In nested CV, a dataset is randomly split in a number of *chunks*. Each of these chunks is kept apart from the remaining chunks once, while the remaining chunks are combined to form a single dataset. A grid search is then applied to this combined dataset to find the optimal hyperparameters with which to predict the test chunk. This process is then repeated for each of the chunks. The predictions of the test chunk will be unbiased since it was not included in the

[4]The LibSVM code used for DAGSVM is the same code as was used in Hsu and Lin (2002) and is available at http://www.csie.ntu.edu.tw/~cjlin/libsvmtools.

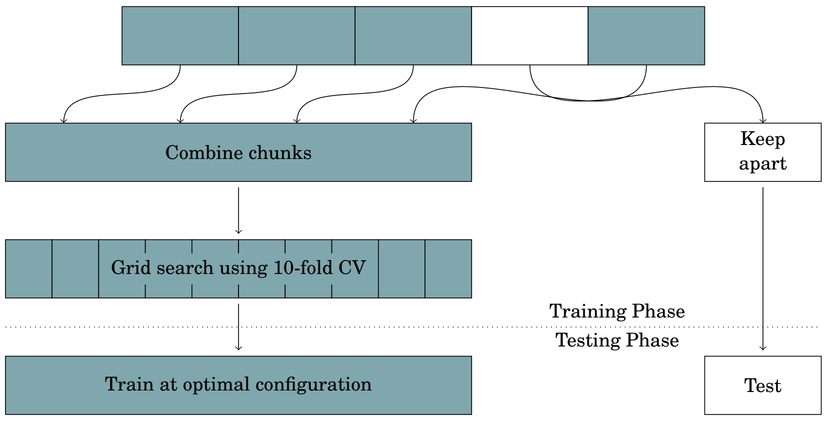


FIGURE 2.5 – *An illustration of nested cross validation. A dataset is initially split in five* chunks. *Each chunk is kept apart once, while a grid search using 10-fold CV is applied to the combined data from the remaining four chunks. The optimal parameters obtained there are then used to train the model one last time and predict the chunk that was kept apart.*

grid search. For this reason, it is argued that this approach is preferred over approaches that simply report maximum accuracy rates obtained during the grid search.

For the experiments 13 datasets were selected from the UCI repository (Bache and Lichman, 2013). The selected datasets and their relevant statistics are shown in Table 2.1. All attributes were rescaled to the interval $[-1, 1]$. The `image segmentation` and `vowel` datasets have a predetermined train and test set and were therefore not used in the nested CV procedure. Instead, a grid search was done on the provided training set for each classifier and the provided test set was predicted at the optimal hyperparameters obtained. For the datasets without a predetermined train/test split, nested CV was used with five initial chunks. Hence, $5 \cdot 11 + 2 = 57$ pairs of independent train and test datasets are obtained.

While running the grid search, it is desirable to remove any fluctuations that may result in an unfair comparison. Therefore, it was ensured that all methods had the same CV split of the training data for the same hyperparameter configuration (specifically, the value of the regularization parameter). In practice, it can occur that a specific CV split is advantageous for one classifier but not for others (either in time or performance). Thus, ideally the grid search would be repeated a number of times with different CV splits, to remove this variation. However, due to the size of the grid search this is considered to be infeasible. Finally, it should be noted here that during the grid search 10-fold cross validation

TABLE 2.1 – *Dataset summary statistics. Datasets with an asterisk have a predetermined test dataset. For these datasets, the number of training instances is denoted for the train and test datasets respectively. The final two columns respectively denote the size of the smallest and the largest class in the dataset.*

Dataset	Instances	Features	Classes	$\min n_k$	$\max n_k$
breast tissue	106	9	6	14	22
iris	150	4	3	50	50
wine	178	13	3	48	71
image segmentation*	210/2100	18	7	30	30
glass	214	9	6	9	76
vertebral	310	6	3	60	150
ecoli	336	8	8	2	143
vowel*	528/462	10	11	48	48
balancescale	625	4	3	49	288
vehicle	846	18	4	199	218
contraception	1473	9	3	333	629
yeast	1484	8	10	5	463
car	1728	6	4	65	1210

was applied in a non-stratified manner, that is, without resampling of small classes.

The following settings were used in the numerical experiments. The regularization parameter was varied on a grid with $\lambda \in \{2^{-18}, 2^{-16}, \ldots, 2^{18}\}$. For GenSVM the grid search was extended with parameters $\kappa \in \{-0.9, 0.5, 5.0\}$ and $p \in \{1.0, 1.5, 2.0\}$. The stopping parameter for the GenSVM majorization algorithm was set at $\epsilon = 10^{-6}$ during the grid search in the training phase and at $\epsilon = 10^{-8}$ for the final model in the testing phase. In addition, two different weight specifications were used for GenSVM: the unit weights with $\rho_i = 1, \forall i$, as well as the group-size correction weights introduced in (2.5). Thus, the grid search consists of 342 configurations for GenSVM and 19 configurations for the other methods. Since nested CV is used for most datasets, it is required to run 10-fold cross validation on a total of 27075 hyperparameter configurations. To enhance the reproducibility of these experiments, the exact predictions made by each classifier for each configuration were stored in a text file.

To run all computations in a reasonable amount of time, the computations were performed on the Dutch National LISA Compute Cluster. A master-worker program was developed using the message passing interface in Python (Dalcín et al., 2005). This allows for efficient use of multiple nodes by successively sending out tasks to worker threads from a single master thread. Since the total training time of a classifier is also of interest, it was ensured that all computations were done on the exact same core type.[5] Furthermore, training time was measured from within the `C` programs, to ensure that only the time needed for the cross validation routine was measured. The total computation time needed to obtain the presented results was about 152 days, using the LISA Cluster this was done in five and a half days wall-clock time.

During the training phase it showed that several of the single machine methods implemented through MSVMpack did not converge to an optimal solution within reasonable amount of time.[6] Instead of limiting the maximum number of iterations of the method, MSVMpack was modified to stop after a maximum of two hours of training time per configuration. This results in 12 minutes of training time per cross validation fold. The solution found after this amount of training time was used for prediction during cross validation. Whenever training was stopped prematurely, this was recorded.[7] Of the 57 training sets, 24 configurations had prematurely stopped training in one or more CV splits for the LLW method, versus 19 for W&W, 9 for MSVM^2, and 2 for C&S (MSVMpack). For the LibSVM methods, 13 optimal configurations for OvA reached the default maximum number of iterations in one or more CV folds, versus 9 for DAGSVM, and 3 for OvO. No early stopping was needed for GenSVM or for LL C&S.

Determining the optimal hyperparameters requires a performance measure on the obtained predictions. For binary classifiers it is common to use either the hitrate or the area under the ROC curve as a measure of classifier performance. The hitrate only measures the percentage of correct predictions of a classifier and has the well known problem that no correction is made for group sizes. For instance, if 90% of the observations of a test set belong to one class, a classifier that always predicts this class has a high hitrate, regardless of its discriminatory power. Therefore, the adjusted Rand index (ARI) is used here as a performance measure (Hubert and Arabie, 1985). The ARI corrects for chance and can therefore more accurately measure discriminatory power of a classifier than the hitrate can. Using the ARI for evaluating supervised learning algorithms has previously been proposed by Santos and Embrechts (2009).

The optimal parameter configurations for each method on each dataset were chosen such that the maximum predictive performance was obtained as measured with the ARI. If multiple configurations obtained the highest performance during

[5]The specific type of core used is the Intel Xeon E5-2650 v2, with 16 threads at a clock speed of 2.6 GHz. At most 14 threads were used simultaneously, reserving one for the master thread and one for system processes.

[6]The default MSVMpack settings were used with a chunk size of 4 for all methods.

[7]For the classifiers implemented through LibSVM very long training times were only observed for the OvA method, however due to the nature of this method it is not trivial to stop the calculations after a certain amount of time. This behavior was observed in about 1% of all configurations tested on all datasets and is therefore considered negligible. Also, for the LibSVM methods it was recorded whenever the maximum number of iterations was reached.

the grid search, the configuration with the smallest training time was chosen. The results on the training data show that during cross validation GenSVM achieved the highest classification accuracy on 41 out of 57 datasets, compared to 15 and 12 for DAG and OvO, respectively. However, these are results on the training datasets and therefore can contain considerable bias. To accurately assess the out-of-sample prediction accuracy the optimal hyperparameter configurations were determined for each of the 57 training sets and the test sets were predicted with these parameters. To remove any variations due to random starts, building the classifier and predicting the test set was repeated five times for each classifier.

Below the simulation results on the small datasets will be evaluated using performance profiles and rank tests. Performance profiles offer a visual representation of classifier performance, while rank tests allow for identification of statistically significant differences between classifiers. For the sake of completeness tables of performance scores and computation times for each method on each dataset are provided in Appendix 2.E. To promote reproducibility of the empirical results, all the code used for the classifier comparisons and all the obtained results will be released through an online repository.

2.6.2 *Performance Profiles*

One way to gain insight in the performance of different classification methods is through *performance profiles* (Dolan and Moré, 2002). A performance profile shows the empirical cumulative distribution function of a classifier on a performance metric.

Let $\mathscr{D}$ denote the set of datasets and let $\mathscr{C}$ denote the set of classifiers. Further, let $p_{d,c}$ denote the performance of classifier $c \in \mathscr{C}$ on dataset $d \in \mathscr{D}$ as measured by the ARI. Now define the performance ratio $v_{d,c}$ as the ratio between the best performance on dataset d and the performance of classifier c on data set d,

$$v_{d,c} = \frac{\max\{p_{d,c} : c \in \mathscr{C}\}}{p_{d,c}}. \tag{2.24}$$

Thus the performance ratio is 1 for the best performing classifier on a dataset and increases for classifiers with a lower performance. Then, the performance profile for classifier c is given by the function

$$P_c(\eta) = \frac{1}{N_D} \left|\{d \in \mathscr{D} : v_{d,c} \leq \eta\}\right|, \tag{2.25}$$

where $N_D = |\mathscr{D}|$ denotes the number of datasets. Thus, the performance profile estimates the probability that classifier c has a performance ratio below η. Note

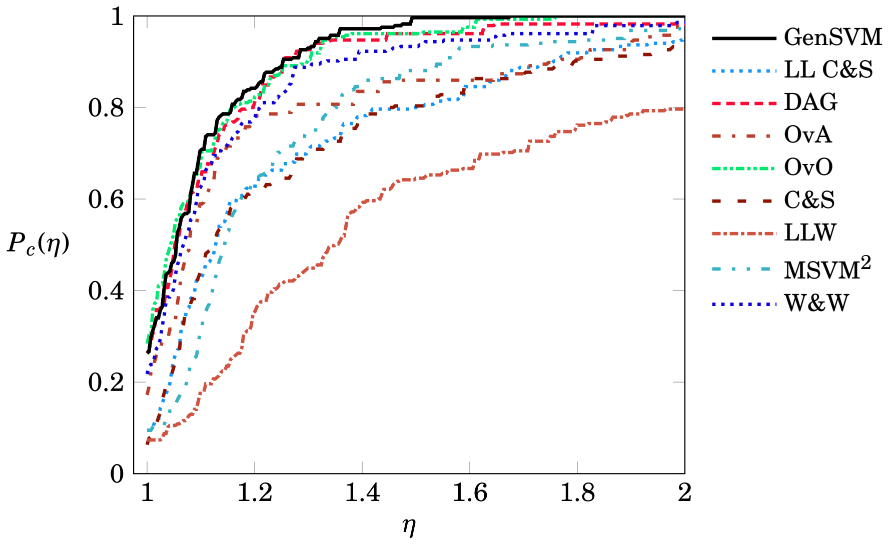


FIGURE 2.6 – *Performance profiles for classification accuracy created from all repetitions of the test set predictions. The methods OvA, C&S, LL C&S, MSVM^2, W&W, and LLW will always have a smaller probability of being within a factor η of the maximum performance than the GenSVM, OvO, or DAG methods.*

that $P_c(1)$ denotes the empirical probability that a classifier achieves the highest performance on a given data set.

Figure 2.6 shows the performance profile for classification accuracy. Estimates of $P_c(1)$ from Figure 2.6 show that there is a 28.42% probability that OvO achieves the optimal performance, versus 26.32% for both GenSVM and DAGSVM. Note that this includes cases where each of these methods achieves the best performance. Figure 2.6 also shows that although there is a small difference in the probabilities of GenSVM, OvO, and DAG within a factor of 1.08 of the best predictive performance, for $\eta \geq 1.08$ GenSVM almost always has the highest probability. It can also be concluded that since the performance profiles of the MSVMpack implementation and the LibLinear implementation of the method of Crammer and Singer (2002a) nearly always overlap, implementation differences have a negligible effect on the classification performance of this method. Finally, the figure shows that OvA and the methods of Lee et al. (2004), Crammer and Singer (2002a), Weston and Watkins (1998), and Guermeur and Monfrini (2011) always have a smaller probability of being within a given factor of the optimal performance than GenSVM, OvO, or DAG do.

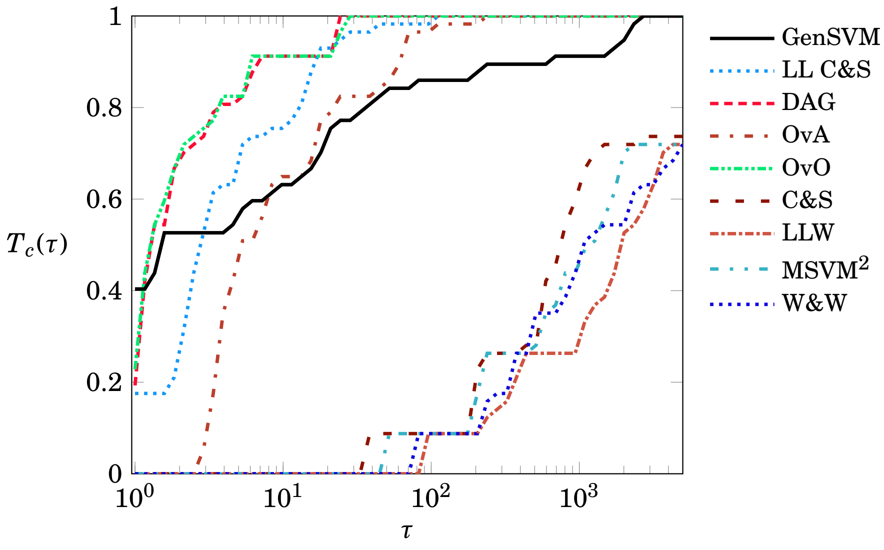


FIGURE 2.7 – *Performance profiles for training time. GenSVM has a priori about 40% chance of requiring the smallest time to perform the grid search on a given dataset. The methods implemented through MSVMpack always have a lower chance of being within a factor τ of the smallest training time than any of the other methods.*

Similarly, a performance profile can be constructed for the training time necessary to do the grid search. Let $t_{d,c}$ denote the total training time for classifier c on dataset d. Next, define the performance ratio for time as

$$w_{d,c} = \frac{t_{d,c}}{\min\{t_{d,c} : c \in \mathscr{C}\}}. \tag{2.26}$$

Since the classifier with the smallest training time has preference the comparison is done with the lowest computation time achieved on a given dataset d. Again, the ratio is 1 when the lowest training time is reached and it increases for higher computation time. Hence, the performance profile for time is defined as

$$T_c(\tau) = \frac{1}{N_D} |\{d \in \mathscr{D} : w_{d,c} \leq \tau\}|. \tag{2.27}$$

The performance profile for time estimates the probability that a classifier c has a time ratio below τ. Again, $T_c(1)$ denotes the fraction of datasets where classifier c achieved the smallest training time among all classifiers.

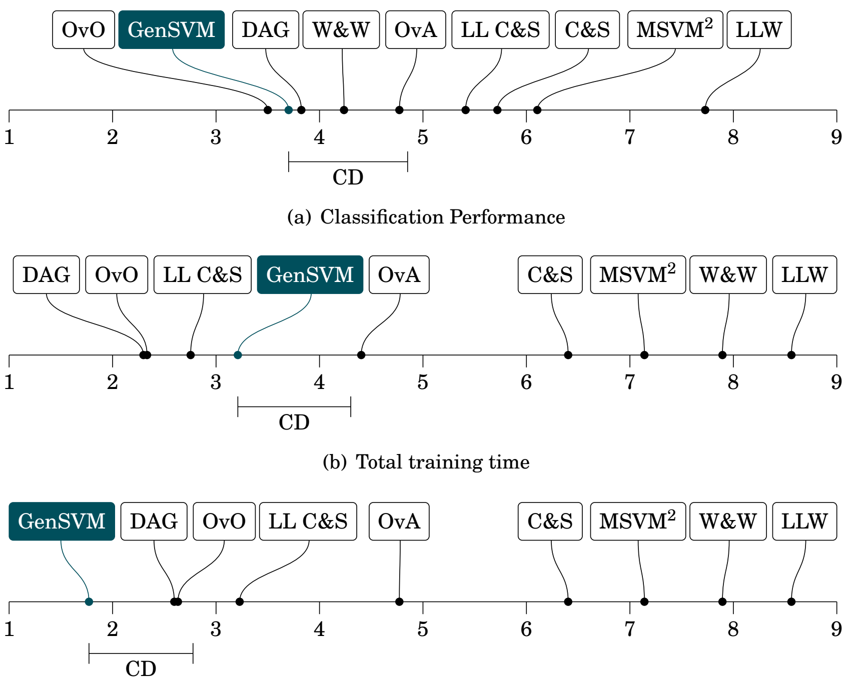
Figure 2.7 shows the performance profile for the time needed to do the grid search. Since large differences in training time were observed, a logarithmic scale is used for the horizontal axis. This performance profile clearly shows that all MSVMpack methods suffer from long computation times. The fastest methods are GenSVM, OvO, and DAG, followed by the LibLinear implementation of C&S. From the value of $T_c(1)$ it is found that GenSVM has the highest probability of being the fastest method for the total grid search, with a probability of 40.35%, versus 22.81% for OvO, 19.30% for DAG, and 17.54% for LibLinear C&S. The other methods never achieve the smallest grid search time. It is important to note here that the grid search for GenSVM is 18 times larger than that of the other methods. These results illustrate the incredible advantage GenSVM has over other methods by using warm starts in the grid search.

In addition to the performance profile, the average computation time per hyperparameter configuration was examined. GenSVM has an average training time of 0.97 seconds per configuration, versus 20.56 seconds for LibLinear C&S, 24.84 seconds for OvO, and 25.03 seconds for DAGSVM. This is a considerable difference, which can be explained again by the use of warm starts in GenSVM (see Section 2.4.3). When the total computation time per dataset is averaged, it is found that GenSVM takes on average 331 seconds per dataset, LibLinear C&S 391 seconds, OvO 472 seconds, and DAG 476 seconds. The difference between DAGSVM and OvO can be attributed to the prediction strategy used by DAGSVM. Thus it can be concluded that on average GenSVM is the fastest method during the grid search, despite the fact it has 18 times more hyperparameters to consider than the other methods.

2.6.3 *Rank Tests*

Following Demšar (2006), ranks are used to investigate significant differences between classifiers. The benefit of using ranks instead of actual performance metrics is that ranks have meaning when averaged across different datasets, whereas average performance metrics do not. Ranks are calculated for the performance as measured by the ARI, the total training time needed to do the grid search, and the average time per hyperparameter configuration. When ties occur fractional ranks are used.

Figure 2.8 shows the average ranks for both classification performance and total and average training time. From Figure 2.8(a) it can be seen that GenSVM is in second place in terms of overall classification performance measured by the ARI. Only OvO has higher performance than GenSVM *on average*. Similarly, Figure 2.8(b) shows the average ranks for the total training time. Here, GenSVM is on average the fourth fastest method for the complete grid search. When looking at the rank plot for the average training time per hyperparameter configuration,



(a) Classification Performance

(b) Total training time

(c) Average training time

FIGURE 2.8 – *Figure* (a) *shows the average ranks for performance,* (b) *shows the average ranks for the total computation time needed for the grid search, and* (c) *shows the ranks for the average time per hyperparameter configuration. It can be seen that GenSVM obtains the second overall rank in predictive performance, fourth overall rank in total training time, and first overall rank in average training time. In all figures, CD shows the critical difference of Holm's procedure. Classifiers beyond this CD differ significantly from GenSVM at the 5% significance level.*

it is clear that the warm starts used during training in GenSVM are very useful as it ranks as the fastest method on this metric, as shown in Figure 2.8(c).

As Demšar (2006) suggests, the Friedman rank test can be used to find significant differences between classifiers (Friedman, 1937, 1940). If r_{cd} denotes the fractional rank of classifier c on dataset d, then with N_C classifiers and N_D datasets the Friedman statistic is given by

$$\chi^2_F = \frac{12N_D}{N_C(N_C+1)}\left[\sum_c R_c^2 - \frac{N_C(N_C+1)^2}{4}\right]. \tag{2.28}$$

Here, $R_c = 1/N_D \sum_d r_{cd}$ denotes the average rank of classifier c. This test statistic is distributed following the χ^2 distribution with $N_C - 1$ degrees of freedom. As

Demšar (2006) notes, Iman and Davenport (1980) showed that the Friedman statistic is undesirably conservative and the F-statistic is to be used instead, which is given by

$$F_F = \frac{(N_D - 1)\chi_F^2}{N_D(N_C - 1) - \chi_F^2}, \tag{2.29}$$

and is distributed following the F-distribution with $N_C - 1$ and $(N_C - 1)(N_D - 1)$ degrees of freedom. Under the null hypothesis of either test there is no significant difference in the performance of any of the algorithms.

Performing the Friedman test on the ranks for classifier performance yields $\chi_F^2 = 116.3$ ($p < 10^{-16}$) and $F_F = 19.2$ ($p = 10^{-16}$). Hence, with both tests the null hypothesis of equal classification accuracy can be rejected. Similarly, for training time the test statistics are $\chi_F^2 = 384.8$ ($p < 10^{-16}$) and $F_F = 302.4$ ($p \approx 10^{-16}$). Therefore, the null hypothesis of equal training time can also be rejected. When significant differences are found through the Friedman test, Demšar (2006) suggests to use Holm's step-down procedure as a post-hoc test, to find which classifiers differ significantly from a chosen reference classifier (Holm, 1979). Here, GenSVM is used as a reference classifier, since comparing GenSVM with existing methods is the main focus of these experiments.

Holm's procedure is based on testing whether the z-statistic comparing classifier i with classifier j is significant, while adjusting for the familywise error rate. Following Demšar (2006), this z-statistic is given by

$$z = (R_{\text{GenSVM}} - R_i)\sqrt{\frac{6N_D}{N_C(N_C + 1)}}, \tag{2.30}$$

where R_{GenSVM} is the average rank of GenSVM and R_i the average rank of another classifier, for $i = 1, \ldots, N_C - 1$. Subsequently, the p-values computed from this statistic are sorted in increasing order, as $p_1 < p_2 < \ldots < p_{N_C - 1}$. Then, the null hypothesis of equal classification accuracy can be rejected if $p_i < \alpha/(N_C - i)$. If for some i the null hypothesis cannot be rejected, all subsequent tests will also fail. By inverting this procedure, a critical difference (CD) can be computed that indicates the minimal difference between the reference classifier and the next best classifier.[8] These critical differences are also illustrated in the rank plots in Figure 2.8.

Using Holm's procedure, it is found that for predictive performance GenSVM significantly outperforms the method of Lee et al. (2004) ($p < 10^{-14}$), the method of Guermeur and Monfrini (2011) ($p = 10^{-6}$), the MSVMpack implementation of Crammer and Singer (2002a) ($p = 4 \cdot 10^{-5}$), and the LibLinear implementation

[8]This is done by taking the smallest value of $\alpha/(N_C - i)$ for which the null hypothesis is rejected, looking up the corresponding z-statistic, and inverting (2.30).

of the same method ($p = 0.0004$) at the 5% significance level. Note that since this last method is included twice these test results are conservative. In terms of total training time GenSVM is significantly faster than all methods implemented through MSVMpack (C&S, W&W, MSVM^2, and LLW) and OvA at the 5% significance level. Recall that the hyperparameter grid for GenSVM is 18 times larger than that of the other methods. When looking at average training time per hyperparameter configuration, GenSVM is significantly faster than all methods except OvO and DAG, at the 1% significance level.

2.6.4 *Large Datasets*

The above results focus on the predictive performance of GenSVM as compared to other multiclass SVM methods. To assess the practicality of GenSVM for large datasets additional simulations were done on three more datasets. The `covtype` dataset ($n = 581016$, $m = 54$, $K = 7$) and the `kddcup99` dataset ($n = 494021$, $m = 116$, $K = 23$) were selected from the UCI repository (Bache and Lichman, 2013).[9] Additionally, the `fars` dataset ($n = 100968$, $m = 338$, $K = 8$) was retrieved from the Keel repository (Alcalá et al., 2010). For large datasets the LibLinear package (Fan et al., 2008) is often used, so the SVM methods from this package were added to the list of alternative methods.[10]

LibLinear includes five different SVM implementations: a coordinate descent algorithm for the ℓ_2-regularized ℓ_1-loss and ℓ_2-loss dual problems (Hsieh et al., 2008), a coordinate descent algorithm for the ℓ_1-regularized ℓ_2-loss SVM (Yuan et al., 2010, Fan et al., 2008), a Newton method for the primal ℓ_2-regularized ℓ_2-loss SVM problem (Lin et al., 2008), and finally a sequential dual method for the multiclass SVM by Crammer and Singer (2002a) introduced by Keerthi et al. (2008). This last method was again included to facilitate a comparison between the implementations of LibLinear and MSVMpack. Note that with the exception of this last method all methods in LibLinear are binary SVMs that implement the one-vs-all strategy.

With the different variants of the linear multiclass SVMs included in LibLinear, a total of 13 methods were considered for these large datasets. Since training of the hyperparameters for each method leads to a high computational

[9]For kddcup99 the 10% training dataset and the corrected test dataset are used here, both available through the UCI repository.

[10]Yet another interesting SVM approach to multiclass classification is the Pegasos method by Shalev-Shwartz et al. (2011). However, the LibLinear package includes five different approaches to SVM, including a fast solver for the method by Crammer and Singer (2002a), which makes it more convenient to include in the list of methods. Moreover, according to the LibLinear documentation (Fan et al., 2008): “LibLinear is competitive or even faster than state of the art linear classifiers such as Pegasos (Shalev-Shwartz et al., 2011) and SVM$^{\text{perf}}$ (Joachims, 2006)”.

Table 2.2 – *Overview of predictive performance on large datasets, as measured by the ARI. Asterisks are used to mark the three best performing methods for each dataset, with three stars denoting the best performing method.*

Package	Method	Covtype	Fars	KDDCup-99
GenSVM	GenSVM	0.3571**	0.8102***	0.9758
LibLinear	L1R-L2L	0.3372	0.8080	0.9762
LibLinear	L2R-L1L (D)	0.3405	0.7995	0.9789
LibLinear	L2R-L2L	0.3383	0.8090**	0.9781
LibLinear	L2R-L2L (D)	0.3393	0.8085*	0.9744
LibLinear	C&S	0.3582***	0.8081	0.9758
LibSVM	DAG		0.8056	0.9809***
LibSVM	OvA		0.7872	0.9800*
LibSVM	OvO		0.8055	0.9804**
MSVMpack	C&S	0.3432*	0.7996	0.9741
MSVMpack	LLW	0.3117	0.7846	0.9660
MSVMpack	MSVM^2	0.3165	0.6567	0.9658
MSVMpack	W&W	0.2848	0.7719	0.6446

burden the nested CV procedure was replaced by a grid search using ten-fold CV on a training set of 80% of the data, followed by out-of-sample prediction on the remaining 20% using the final model. The kddcup99 dataset comes with a separate test dataset of 292302 instances, so this was used for the out-of-sample predictions. The grid search on the training set used the same hyperparameter configurations as for the small datasets above, with 342 configurations for GenSVM and 19 configurations for the other methods. The only difference was that for GenSVM $\epsilon = 10^{-9}$ was used when training the final model. To accelerate the GenSVM computations, support for sparse matrices was added.

Due to the large dataset sizes, many methods had trouble converging within a reasonable amount of time. Therefore, total computation time was limited to five hours per hyperparameter configuration per method, both during CV and when training the final model. Where possible this limitation was included in the main optimization routine of each method, such that training was stopped when convergence was reached or when more than five hours had passed. Additionally, for all methods the CV procedure was stopped prematurely if more than five hours had passed after completion of a fold. In this case, cross validation performance is only measured for the folds that were completed. These computations were again performed on the Dutch National LISA Compute Cluster.

Table 2.2 shows the out-of-sample predictive performance of the different MSVMs on the large datasets. It can be seen that GenSVM is the best performing method on the fars dataset and the second best method on the covtype dataset,

Table 2.3 – *Overview of training time for each of the large datasets. The average training time per hyperparameter configuration is also shown. All values are reported in seconds. For LibSVM the full grid search could never be completed and results are averaged only over the finished configurations.*

		Covtype		Fars		KDDCup-99	
Package	Method	Total	Mean	Total	Mean	Total	Mean
GenSVM	GenSVM	166949	488	131174	384	1768303	5170
LibLinear	L1R-L2L	69469	3656	4199	221	34517	1817
LibLinear	L2R-L1L (D)	134908	7100	6995	368	16347	860
LibLinear	L2R-L2L	4168	219	746	39	3084	162
LibLinear	L2R-L2L (D)	159781	8410	7897	416	16974	893
LibLinear	C&S	166719	8775	124764	6567	5425	286
LibSVM	DAG	80410	40205	81557	8156	61111	3595
LibSVM	OvA	77335	77335	54965	18322	73871	12312
LibSVM	OvO	140826	46942	84580	8458	81023	4501
MSVMpack	C&S	350397	18442	351664	18509	365733	19249
MSVMpack	LLW	370790	19515	380943	20050	361329	19017
MSVMpack	MSVM2	370736	19512	346140	18218	353479	18604
MSVMpack	W&W	367245	19329	344880	18152	367685	19352

just after LL C&S. The LibSVM methods outperform the other methods on the `kddcup99` dataset, with DAGSVM having the highest performance. No results are available for LibSVM for the `covtype` dataset because convergence could not be reached within the five hour time limit during the test phase.

Results on the computation time are reported in Table 2.3. The ℓ_2-regularized ℓ_2-loss method by Lin et al. (2008) is clearly the fastest method. However, for the `covtype` dataset GenSVM total training time is competitive with some of the other LibLinear methods and outperforms these methods in terms of average training time. For the `fars` dataset the average training time of GenSVM is also competitive with some of the LibLinear methods, most notably the method by Crammer and Singer (2002a). The MSVMpack methods seem to be infeasible for such large datasets, as computations were stopped by the five hour time limit for almost all hyperparameter configurations. Early stopping was also needed for the LibLinear implementation of C&S on the `covtype` and `fars` datasets, and for the LibSVM methods on all datasets. For GenSVM, early stopping was only needed for the `kddcup99` dataset, which explains the high total computation time there. Especially on these large datasets the advantage of using warm starts in GenSVM is visible: training time was less than 30 seconds in 30% of hyperparameters on `fars`, 23% on `covtype`, and 11% on `kddcup99`.

2.7 Discussion

A generalized multiclass support vector machine has been introduced, called GenSVM. The method is general in the sense that it subsumes three multiclass

SVMs proposed in the literature and it is flexible due to several different weighting options. The simplex encoding of the multiclass classification problem used in GenSVM is intuitive and has an elegant geometrical interpretation. An iterative majorization algorithm has been derived to minimize the convex GenSVM loss function in the primal. This primal optimization approach has computational advantages due to the possibility to use warm starts and because it can be easily understood. The ability to use warm starts contributes to small training time during cross validation in a grid search and allows GenSVM to perform competitively on large datasets.

Rigorous computational tests of linear multiclass SVMs on small datasets show that GenSVM significantly outperforms three existing multiclass SVMs (four implementations) on predictive performance at the 5% significance level. On this metric, GenSVM is the second-best performing method overall and the best method among single-machine multiclass SVMs, although the difference with the method of Weston and Watkins (1998) could not be shown to be statistically significant. GenSVM outperforms five other methods on total training time and has the smallest total training time when averaged over all datasets, despite the fact that its grid of hyperparameters is 18 times larger than that of other methods. Due to the possibility of warm starts it also has the smallest average training time per hyperparameter and significantly outperforms all but two alternative methods in this regard at the 1% significance level. For the large datasets, it was found that GenSVM still achieves high classification accuracy and that total training time remains manageable due to the warm starts. In practice, the number of hyperparameters could be reduced if smaller training time is desired. Since GenSVM outperforms existing methods on a number of datasets and achieves fast training time it is a worthwhile addition to the collection of methods available to the practitioner.

In the comparison tests MSVMpack (Lauer and Guermeur, 2011) was used to access four single machine multiclass SVMs proposed in the literature. A big advantage of using this library is that it allows for a single straightforward `C` implementation, which greatly reduces the programming effort needed for the comparisons. However, as is noted in the MSVMpack documentation, slight differences exist between MSVMpack and method-specific implementations. For instance, on small datasets MSVMpack can be slower, due to working set selection and shrinking procedures in other implementations. However, classification performance is comparable between MSVMpack and method-specific implementations, as was verified by adding the LibLinear implementation of the method of Crammer and Singer (2002a) to the list of alternative methods. Thus, it is argued that the results for predictive accuracy presented above are accurate regardless of implementation, but small differences can exist for training time when other implementations for single machine MSVMs are used.

Another interesting conclusion that can be drawn from the experimental results is that the one-vs-all method never performs as good as one-vs-one, DAGSVM, or GenSVM. In fact, the profile plot in Figure 2.6 shows that OvA always has a smaller probability of obtaining the best classification performance as either of these three methods. These results are also reflected in the classification accuracy of the LibLinear methods on the large dataset. In the literature, the paper by Rifkin and Klautau (2004) is often cited as evidence that OvA performs well (see for instance Keerthi et al., 2008). However, the simulation results in this paper suggest that OvA is in fact inferior to OvO, DAG, and GenSVM.

This chapter was focused on linear multiclass SVMs. An obvious extension is to incorporate nonlinear multiclass SVMs through kernels. Due to the large number of datasets and the long training time the numerical experiments were limited to linear multiclass SVM. Nonlinear classification through kernels can be achieved by linear methods through a preprocessing step of an eigendecomposition on the kernel matrix, which is a process of the order $O(n^3)$. In this case, GenSVM will benefit from precomputing kernels before starting the grid search, or using a larger stopping criterion in the IM algorithm by increasing ϵ in Algorithm 2.1. In addition, approximations can be done by using rank approximated kernel matrices, such as the Nyström method proposed by Williams and Seeger (2001). Such enhancements are considered topics for further research.

Finally, the potential of using GenSVM in an online setting is recognized. Since the solution can be found quickly when a warm-start is used, GenSVM may be useful in situations where new instances have to be predicted at a certain moment, while the true class label arrives later. Then, re-estimating the GenSVM solution can be done as soon as the true class label of an object arrives and a previously known solution can be used as a warm start. It is expected that in this scenario only a few iterations of the IM algorithm are needed to arrive at a new optimal solution. This, too, is considered a subject for further research.

Acknowledgements

The computational experiments of this work were performed on the Dutch National LISA Compute Cluster and supported by the Dutch National Science Foundation (NWO). The authors thank SURFsara (www.surfsara.nl) for the support in using the LISA cluster.

Appendices

2.A Simplex Coordinates

The simplex used in the formulation of the GenSVM loss function is a regular K-simplex in $\mathbb{R}^{K-1}$ with distance 1 between each pair of vertices, which is centered at the origin. Since these requirements alone do not uniquely define the simplex coordinates in general, it will be chosen such that at least one of the vertices lies on an axis. The 2-simplex in $\mathbb{R}^1$ is uniquely defined with the coordinates $-\frac{1}{2}$ and $+\frac{1}{2}$. Using these requirements, it is possible to define a recursive formula for $\mathbf{U}_K$, the simplex coordinate matrix of the K-simplex in $\mathbb{R}^{K-1}$:

$$\mathbf{U}_K = \begin{bmatrix} \mathbf{U}_{K-1} & \mathbf{1}t \\ \mathbf{0}' & s \end{bmatrix}, \qquad \text{with} \qquad \mathbf{U}_2 = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}. \tag{2.31}$$

Note that the matrix $\mathbf{U}_K$ has K rows and $K-1$ columns. Since the simplex is centered at zero it holds that the elements in each column sum to 0, implying that $s = -(K-1)t$. Denote by $\mathbf{u}'_i$ the i-th row of $\mathbf{U}_K$ and by $\tilde{\mathbf{u}}'_i$ the i-th row of $\mathbf{U}_{K-1}$, then it follows from the edge length requirement that,

$$\|\mathbf{u}'_i - \mathbf{u}'_K\|^2 = \|\tilde{\mathbf{u}}'_i - \mathbf{0}' + t - s\|^2 = \|\tilde{\mathbf{u}}'_i\|^2 + (t-s)^2 = 1, \quad \forall i \neq K. \tag{2.32}$$

From the requirement of equal distance from each vertex to the origin it follows that $\|\mathbf{u}'_i\|^2 = \|\mathbf{u}'_K\|^2$ and $\|\tilde{\mathbf{u}}'_i\|^2 + t^2 = s^2$, $\forall i \neq K$. Combining these two expressions yields the equation $2s^2 - 2st - 1 = 0$. Substituting $s = -(K-1)t$ and choosing $s > 0$ and $t < 0$ gives:

$$t = \frac{-1}{\sqrt{2K(K-1)}}, \quad s = \frac{K-1}{\sqrt{2K(K-1)}}. \tag{2.33}$$

Note that using $K = 2$ in these expressions gives $t = -\frac{1}{2}$ and $s = \frac{1}{2}$, as expected. The recursive relationship defined above then reveals that the first $K-1$ elements in column $K-1$ of the matrix are equal to t, and the K-th element in column $K-1$ is equal to s. This can then be generalized for an element u_{kl} in row k and column l of $\mathbf{U}_K$, yielding the expression given in (2.1).

2.B Details of Iterative Majorization

In this section a brief introduction to iterative majorization is given, following the description of Voss and Eckhardt (1980). The section concludes with a note on step doubling, a common technique to speed up quadratic majorization algorithms.

Given a continuous function $f : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} \subseteq \mathbb{R}^d$, construct a *majorization function* $g(x, \overline{x})$ such that

$$f(\overline{x}) = g(\overline{x}, \overline{x}), \tag{2.34}$$

$$f(x) \leq g(x, \overline{x}) \text{ for all } x \in \mathcal{X}, \tag{2.35}$$

with $\overline{x} \in \mathcal{X}$ a so-called *supporting point*. In general, the majorization function is constructed such that its minimum can easily be found, for instance by choosing it to be quadratic in x. If $f(x)$ is differentiable at the supporting point, the above conditions imply $\nabla f(\overline{x}) = \nabla g(\overline{x}, \overline{x})$. The following procedure can now be used to find a stationary point of $f(x)$,

1. Let $\overline{x} = x_0$, with x_0 a random starting point.
2. Minimize $g(x, \overline{x})$ with respect to x, such that $x^+ = \arg\min g(x, \overline{x})$.
3. If $f(\overline{x}) - f(x^+) < \epsilon f(x^+)$ stop, otherwise let $\overline{x} = x^+$ and go to step 2.

In this algorithm ϵ is a small constant. Note that $f(x)$ must be bounded from below on $\mathcal{X}$ for the algorithm to converge. In fact, the following *sandwich inequality* can be derived (De Leeuw, 1993),

$$f(x^+) \leq g(x^+, \overline{x}) \leq g(\overline{x}, \overline{x}) = f(\overline{x}). \tag{2.36}$$

This inequality shows that if $f(x)$ is bounded from below the iterative majorization algorithm achieves global convergence to a stationary point of the function (Voss and Eckhardt, 1980). The iterative majorization algorithm is illustrated in Figure 2.9, where the majorization functions are shown as a quadratic function. As can be seen from the illustration, the sequence of supporting points $\{x_r\}$ converges to the stationary point x^* of the function $f(x)$. In practical situations, this convergence is to a local minimum of $f(x)$.

For quadratic majorization the number of iterations can often be reduced by using a technique known as *step doubling* (De Leeuw and Heiser, 1980).

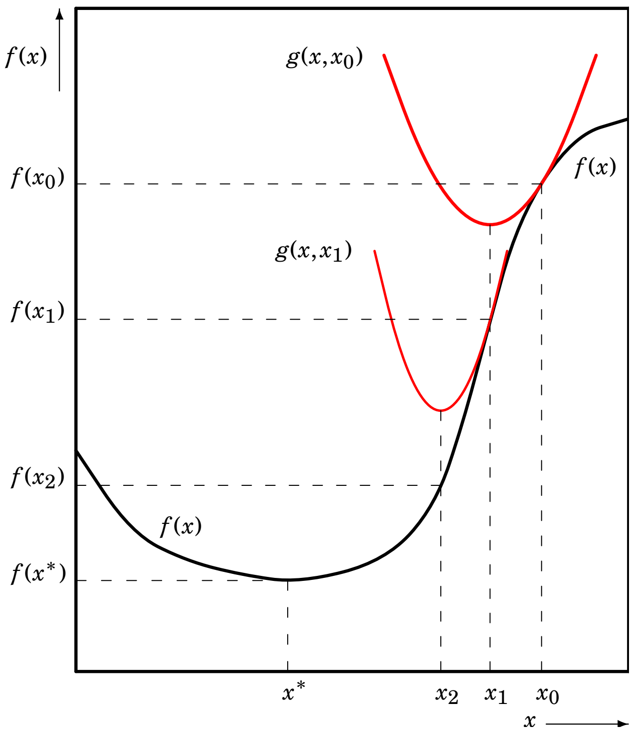


FIGURE 2.9 – *One-dimensional graphical illustration of the iterative majorization algorithm, adapted from De Leeuw (1988). The minimum of a majorization function $g(x,x_r)$ provides the supporting point for the next majorization function $g(x,x_{r+1})$. The sequence of supporting points $\{x_r\}$ converges towards the stationary point x^* if $f(x)$ is bounded from below, as is the case here.*

Step doubling reduces the number of iterations by using $\overline{x} = x_{r+1} = 2x^+ - x_r$ as the next supporting point in Step 3 of the algorithm, instead of $\overline{x} = x_{r+1} = x^+$. Intuitively, step doubling can be understood as stepping over the minimum of the majorization function to the point lying directly "opposite" the supporting point $\overline{x}$ (see also Figure 2.9). Note that the guaranteed descent of the IM algorithm still holds when using step doubling, since $f(2x^+ - \overline{x}) \leq g(2x^+ - \overline{x}, \overline{x}) = g(\overline{x}, \overline{x}) = f(\overline{x})$. In practice, step doubling reduces the number of iterations by half. A caveat of using step doubling is that the distance to the stationary point can be increased if the initial point is far from this point. Therefore, in practical applications, a burn-in should be used before step doubling is applied.

2.C Huber Hinge Majorization

In this appendix, the majorization function will be derived of the Huber hinge error raised to the power p. Thus, a quadratic function $g(x,\overline{x}) = ax^2 - 2bx + c$ is required, which is a majorization function of

$$f(x) = h^p(x) = \begin{cases} \left(1 - x - \frac{\kappa+1}{2}\right)^p & \text{if } x \leq -\kappa \\ \frac{1}{(2(\kappa+1))^p}(1-x)^{2p} & \text{if } x \in (-\kappa, 1] \\ 0 & \text{if } x > 1, \end{cases} \tag{2.37}$$

with $p \in [1,2]$. Each piece of $f(x)$ provides a possible region for the supporting point $\overline{x}$. These regions will be treated separately, starting with $\overline{x} \in (-\kappa, 1]$.

Since the majorization function must touch $f(x)$ at the supporting point, we can solve $f(\overline{x}) = g(\overline{x},\overline{x})$ and $f'(\overline{x}) = g'(\overline{x},\overline{x})$ for b and c to find

$$b = a\overline{x} + \frac{p}{1-\overline{x}}\left(\frac{1-\overline{x}}{\sqrt{2(\kappa+1)}}\right)^{2p}, \tag{2.38}$$

$$c = a\overline{x}^2 + \left(1 + \frac{2p\overline{x}}{1-\overline{x}}\right)\left(\frac{1-\overline{x}}{\sqrt{2(\kappa+1)}}\right)^{2p}, \tag{2.39}$$

whenever $\overline{x} \in (-\kappa, 1]$. Note that since $p \in [1,2]$ the function $f(x)$ can become proportional to a fourth power on the interval $x \in (-\kappa, 1]$. The upper bound of the second derivative of $f(x)$ on this interval is reached at $x = -\kappa$. Equating $f''(-\kappa)$ to $g''(-\kappa,\overline{x}) = 2a$ and solving for a yields

$$a = \tfrac{1}{4}p(2p-1)\left(\frac{\kappa+1}{2}\right)^{p-2}. \tag{2.40}$$

Figure 2.10(a) shows an illustration of the majorization function when $\overline{x} \in (-\kappa, 1]$.

For the interval $\overline{x} \leq -\kappa$ the following expressions are found for b and c using similar reasoning as above

$$b = a\overline{x} + \tfrac{1}{2}p\left(1 - \overline{x} - \frac{\kappa+1}{2}\right)^{p-1}, \tag{2.41}$$

$$c = a\overline{x}^2 + p\overline{x}\left(1 - \overline{x} - \frac{\kappa+1}{2}\right)^{p-1} + \left(1 - \overline{x} - \frac{\kappa+1}{2}\right)^p. \tag{2.42}$$

To obtain the largest possible majorization step it is desired that the minimum of the majorization function is located at $x \geq 1$, such that $g(x_{min},\overline{x}) = 0$. This

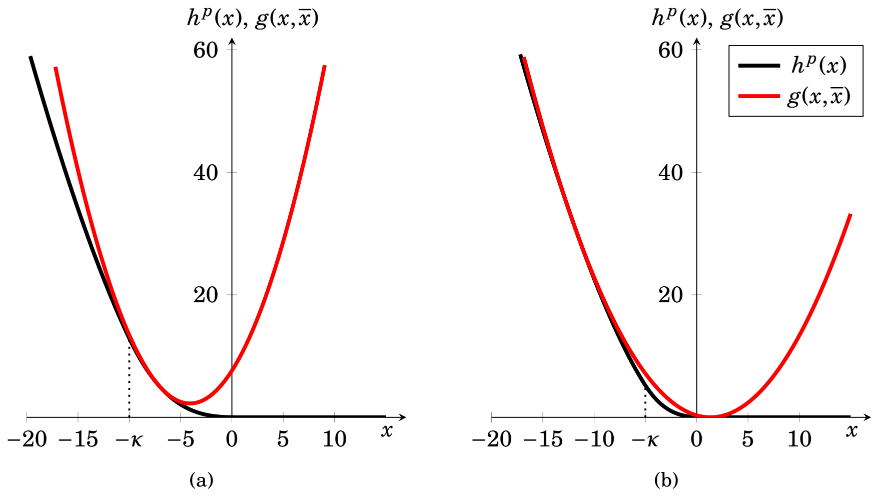


FIGURE 2.10 – *Graphical illustration of the majorization of the function $f(x) = h^p(x)$. Figure* (a) *shows the case where $\overline{x} \in (-\kappa, 1]$, whereas* (b) *shows the case where $\overline{x} \leq (p + \kappa - 1)/(p - 2)$. In both cases $p = 1.5$. It can be seen that in* (b) *the minimum of the majorization function lies at $x > 1$, such that the largest possible majorization step is obtained.*

requirement yields $c = b^2/a$, which gives

$$a = \tfrac{1}{4}p^2\left(1 - \overline{x} - \frac{\kappa + 1}{2}\right)^{p-2}. \tag{2.43}$$

Note however that due to the requirement that $f(x) \leq g(x, \overline{x})$ for all $x \in \mathbb{R}$, this majorization is not valid for all values of $\overline{x}$. Solving the requirement for the minimum of the majorization function, $g(x_{min}, \overline{x}) = 0$ for $\overline{x}$ yields

$$\overline{x} \leq \frac{p + \kappa - 1}{p - 2}. \tag{2.44}$$

Thus, if $\overline{x}$ satisfies this condition, (2.43) can be used for a, whereas for cases where $\overline{x} \in ((p+\kappa-1)/(p-2), -\kappa]$, the value of a given in (2.40) can be used. Figure 2.10(b) shows an illustration of the case where $\overline{x} \leq (p + \kappa - 1)/(p - 2)$.

Next, a majorization function for the interval $\overline{x} > 1$ is needed. Since it has been derived that for the interval $\overline{x} \leq (p+\kappa-1)/(p-2)$ the minimum of the majorization function lies at $x \geq 1$, symmetry arguments can be used to derive the majorization function for $\overline{x} > 1$, and ensure that it is also tangent at $x = (p\overline{x} + \kappa - 1)/(p - 2)$. This

TABLE 2.4 – *Overview of quadratic majorization coefficients for different pieces of $h^p(x)$, depending on $\overline{x}$.*

Region	a	b	c
$\overline{x} \leq \frac{p+\kappa-1}{p-2}$	(2.43)	(2.41)	(2.42)
$\overline{x} \in \left(\frac{p+\kappa-1}{p-2}, -\kappa\right]$	(2.40)	(2.41)	(2.42)
$\overline{x} \in (-\kappa, 1]$	(2.40)	(2.38)	(2.39)
$\overline{x} > 1$, $p \neq 2$	(2.45)	(2.46)	(2.47)
$\overline{x} > 1$, $p = 2$	(2.40)	$a\overline{x}$	$a\overline{x}^2$

yields the coefficients

$$a = \tfrac{1}{4}p^2\left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p-2}, \tag{2.45}$$

$$b = a\left(\frac{p\overline{x}+\kappa-1}{p-2}\right)+\tfrac{1}{2}\left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p-1}, \tag{2.46}$$

$$\begin{aligned} c = {} & a\left(\frac{p\overline{x}+\kappa-1}{p-2}\right)^2 + p\left(\frac{p\overline{x}+\kappa-1}{p-2}\right)\left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p-1} \\ & + \left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p}. \end{aligned} \tag{2.47}$$

Finally, observe that some of the above coefficients are invalid if $p = 2$. However, since the upper bound on the interval $\overline{x} \in (-\kappa, 1]$ given in (2.40) is still valid if $p = 2$, it is possible to do a separate derivation with this value for a to find for $\overline{x} > 1$, $b = a\overline{x}$ and $c = a\overline{x}^2$. For the other regions the previously derived coefficients still hold. Table 2.4 gives an overview of the various coefficients depending on the location of $\overline{x}$.

2.D KERNELS IN GENSVM

To included kernels in GenSVM it is necessary to perform a preprocessing step on the kernel matrix and a postprocessing step on the obtained parameters. Let $k : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^+$ denote a positive definite kernel satisfying Mercer's theorem, and let $\mathscr{H}_k$ denote the corresponding reproducing kernel Hilbert space. Furthermore, define a feature mapping $\phi : \mathbb{R}^m \to \mathscr{H}_k$ as $\phi(\mathbf{x}) = k(\mathbf{x}, \cdot)$, such that by the reproducing property of k it holds that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathscr{H}_k}$.

Using this, the kernel matrix $\mathbf{K}$ is defined as the $n \times n$ matrix with elements $k(\mathbf{x}_i, \mathbf{x}_j)$ on the i-th row and j-th column. Thus, if $\mathbf{\Phi}$ denotes the $n \times l$ matrix with rows $\phi(\mathbf{x}_i)$ for $i = 1, \ldots, n$ and $l \in [1, \infty]$, then $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}'$. Note that it depends on the chosen kernel whether $\mathbf{\Phi}$ is finite dimensional. However, the rank of $\mathbf{\Phi}$ can still be determined through $\mathbf{K}$, since $r = \text{rank}(\mathbf{\Phi}) = \text{rank}(\mathbf{K}) \leq \min(n, l)$.

Now, let the reduced singular value decomposition of $\mathbf{\Phi}$ be given by

$$\mathbf{\Phi} = \mathbf{P\Sigma Q}', \tag{2.48}$$

where $\mathbf{P}$ is $n \times r$, $\mathbf{\Sigma}$ is $r \times r$, and $\mathbf{Q}$ is $l \times r$. Note that here, $\mathbf{P}'\mathbf{P} = \mathbf{I}_r$, $\mathbf{Q}'\mathbf{Q} = \mathbf{I}_r$, and $\mathbf{\Sigma}$ is diagonal. Under the mapping $\mathbf{X} \to \mathbf{\Phi}$ the simplex space vectors become

$$\mathbf{S} = \mathbf{\Phi W} + \mathbf{1t}' = \mathbf{P\Sigma Q}'\mathbf{W} + \mathbf{1t}' = \mathbf{MQ}'\mathbf{W} + \mathbf{1t}'. \tag{2.49}$$

Here $\mathbf{W}$ is $l \times (K-1)$ to correspond to the dimensions of $\mathbf{\Phi}$, and the $n \times r$ matrix $\mathbf{M} = \mathbf{P\Sigma}$ has been introduced. In general $\mathbf{W}$ cannot be determined, since l might be infinite. This problem can be solved as follows. Decompose $\mathbf{W}$ in two parts, $\mathbf{W} = \mathbf{W}_1 + \mathbf{W}_2$, where $\mathbf{W}_1$ is in the linear space of $\mathbf{Q}$ and $\mathbf{W}_2$ is orthogonal to that space, such that $\mathbf{W}_1 = \mathbf{QQ}'\mathbf{W}$ and $\mathbf{W}_2 = (\mathbf{I}_l - \mathbf{QQ}')\mathbf{W}$. Then it follows that

$$\begin{aligned}
\mathbf{S} &= \mathbf{MQ}'\mathbf{W} + \mathbf{1t}' \\
&= \mathbf{MQ}'(\mathbf{W}_1 + \mathbf{W}_2) + \mathbf{1t}' \\
&= \mathbf{MQ}'(\mathbf{W}_1 + (\mathbf{I}_l - \mathbf{QQ}')\mathbf{W}) + \mathbf{1t}' \\
&= \mathbf{MQ}'\mathbf{W}_1 + \mathbf{M}(\mathbf{Q}' - \mathbf{Q}'\mathbf{QQ}')\mathbf{W} + \mathbf{1t}' \\
&= \mathbf{MQ}'\mathbf{W}_1 + \mathbf{M}(\mathbf{Q}' - \mathbf{Q}')\mathbf{W} + \mathbf{1t}' \\
&= \mathbf{MQ}'\mathbf{W}_1 + \mathbf{1t}',
\end{aligned}$$

where it has been used that $\mathbf{Q}'\mathbf{Q} = \mathbf{I}_r$. If the penalty term of the GenSVM loss function is considered, it is found that

$$P_\lambda(\mathbf{W}) = \lambda \operatorname{tr} \mathbf{W}'\mathbf{W} = \lambda \operatorname{tr} \mathbf{W}_1'\mathbf{W}_1 + \lambda \operatorname{tr} \mathbf{W}_2'\mathbf{W}_2, \tag{2.50}$$

since

$$\mathbf{W}_1'\mathbf{W}_2 = \mathbf{W}'\mathbf{QQ}'(\mathbf{I}_l - \mathbf{QQ}')\mathbf{W} = \mathbf{W}'\mathbf{QQ}'\mathbf{W} - \mathbf{W}'\mathbf{QQ}'\mathbf{W} = \mathbf{O}. \tag{2.51}$$

Here again it has been used that $\mathbf{Q}'\mathbf{Q} = \mathbf{I}_r$, and $\mathbf{O}$ is defined as a $(K-1) \times (K-1)$ dimensional matrix of zeroes. Note that the penalty term depends on $\mathbf{W}_2$ whereas the simplex vectors $\mathbf{S}$ do not. Therefore, at the optimal solution it is required that $\mathbf{W}_2$ is zero, to minimize the loss function.

Since $\mathbf{W}_1$ is still $l \times (K-1)$ dimensional with l possibly infinite, consider the substitution $\mathbf{W}_1 = \mathbf{Q\Omega}$, with $\mathbf{\Omega}$ an $r \times (K-1)$ matrix. Then the penalty term is

$$P_\lambda(\mathbf{W}_1) = \lambda \operatorname{tr} \mathbf{W}_1'\mathbf{W}_1 = \lambda \operatorname{tr} \mathbf{\Omega}'\mathbf{Q}'\mathbf{Q\Omega} = \lambda \operatorname{tr} \mathbf{\Omega}'\mathbf{\Omega} = P_\lambda(\mathbf{\Omega}). \tag{2.52}$$

Note also that

$$\mathbf{S} = \mathbf{MQ}'\mathbf{W}_1 + \mathbf{1t}' = \mathbf{MQ}'\mathbf{Q\Omega} + \mathbf{1t}' = \mathbf{M\Omega} + \mathbf{1t}'. \tag{2.53}$$

The question remains on how to determine the matrices $\mathbf{P}$ and $\mathbf{\Sigma}$, given that the matrix $\mathbf{\Phi}$ cannot be determined explicitly. These matrices can be determined by the eigendecomposition of $\mathbf{K}$, where $\mathbf{K} = \mathbf{P\Sigma}^2\mathbf{P}'$. In the case where $r < n$, $\mathbf{\Sigma}^2$ contains only the first r eigenvalues of $\mathbf{K}$, and $\mathbf{P}$ the corresponding r columns. Hence, if $\mathbf{K}$ is not of full rank, a dimensionality reduction is achieved in $\mathbf{\Omega}$. The complexity of finding the eigendecomposition of the kernel matrix is $O(n^3)$.

Since the distances $q_i^{(kj)}$ in the GenSVM loss function can be written as $q_i^{(kj)} = \mathbf{s}_i'\boldsymbol{\delta}_{kj}$ it follows that the errors can again be calculated in this formulation. Finally, to predict the simplex space vectors of a test set $\mathbf{X}_2$ the following is used. Let $\mathbf{\Phi}_2$ denote the feature space mapping of $\mathbf{X}_2$, then

$$\begin{aligned}
\mathbf{S}_2 &= \mathbf{\Phi}_2\mathbf{W}_1 + \mathbf{1t}' \\
&= \mathbf{\Phi}_2\mathbf{Q\Omega} + \mathbf{1t}' \\
&= \mathbf{\Phi}_2\mathbf{Q\Sigma P}'\mathbf{P\Sigma}^{-1}\mathbf{\Omega} + \mathbf{1t}' \\
&= \mathbf{\Phi}_2\mathbf{\Phi}'\mathbf{P\Sigma}^{-1}\mathbf{\Omega} + \mathbf{1t}' \\
&= \mathbf{K}_2\mathbf{P\Sigma}^{-1}\mathbf{\Omega} + \mathbf{1t}' \\
&= \mathbf{K}_2\mathbf{M\Sigma}^{-2}\mathbf{\Omega} + \mathbf{1t}',
\end{aligned}$$

where $\mathbf{K}_2 = \mathbf{\Phi}_2\mathbf{\Phi}'$ is the kernel matrix between the test set and the training set, and it was used that $\mathbf{\Sigma P}'\mathbf{P\Sigma}^{-1} = \mathbf{I}_r$, and $\mathbf{\Phi}' = \mathbf{Q\Sigma P}'$ by definition.

With the above expressions for $\mathbf{S}$ and $P_\lambda(\mathbf{\Omega})$, it is possible to derive the majorization function of the loss function for the nonlinear case. The first order conditions can then again be determined, which yields the following system

$$\left(\begin{bmatrix}\mathbf{1}' \\ \mathbf{M}'\end{bmatrix} \mathbf{A} \begin{bmatrix}\mathbf{1} & \mathbf{M}\end{bmatrix} + \lambda \begin{bmatrix} 0 & \mathbf{0}' \\ \mathbf{0} & \mathbf{I}_r \end{bmatrix}\right) \begin{bmatrix}\mathbf{t}' \\ \mathbf{\Omega}\end{bmatrix} = \begin{bmatrix}\mathbf{1}' \\ \mathbf{M}'\end{bmatrix} \mathbf{A} \begin{bmatrix}\mathbf{1} & \mathbf{M}\end{bmatrix} \begin{bmatrix}\bar{\mathbf{t}}' \\ \overline{\mathbf{\Omega}}\end{bmatrix} + \begin{bmatrix}\mathbf{1}' \\ \mathbf{M}'\end{bmatrix} \mathbf{B}. \tag{2.54}$$

This system is analogous to the system solved in linear GenSVM. In fact, it can be shown that by writing $\mathbf{Z} = [\mathbf{1}\,\mathbf{M}]$ and $\mathbf{V} = [\mathbf{t}'\ \mathbf{\Omega}]'$, this system is equivalent to (2.23). This property is very useful for the implementation of GenSVM, since nonlinearity can be included by simply adding a pre- and post-processing step to the existing GenSVM algorithm.

2.E Additional Simulation Results

Tables 2.5 and 2.6 respectively show the predictive accuracy rates and ARI scores on each dataset averaged over each of the five test folds. For readability all scores are rounded to four decimal digits, however identifying the classifier with the highest score was done on the full precision scores. As can be seen, the choice of performance metric has an effect on which classification method has the highest classification performance. Regardless of the performance metric the tables show that MSVM^2 and W&W never achieve the maximum classification performance on a dataset. Note that conclusions drawn from tables of performance scores are quite limited and the results presented in Section 2.6 provide more insight into the performance of the various classifiers.

Table 2.7 shows the computation time averaged over the five nested CV folds for each dataset and each method. In the grid search GenSVM considered 342 hyperparameter configurations versus 19 configurations for the other methods. Despite this difference GenSVM outperformed the other methods on five datasets, DAG outperformed other methods on four datasets, OvO on two, and LibLinear C&S was fastest on the remaining two datasets. To illustrate the effect of the larger grid search in GenSVM on the computation time, Table 2.8 shows the average computation time per hyperparameter configuration. This table shows that GenSVM is faster than other methods on nine out of thirteen datasets, which illustrates the influence of warm starts in the GenSVM grid search.

TABLE 2.5 – *Predictive accuracy rates for each of the classification methods on all datasets. All numbers are out-of-sample prediction accuracies averaged over the five independent test folds. Maximum scores per dataset are determined on the full precision scores and are underlined.*

Dataset	GenSVM	LL C&S	DAG	OvA	OvO	C&S	LLW	MSVM^2	W&W
balancescale	0.9168	0.8883	0.9168	0.8928	0.9168	0.8922	0.8701	0.8714	0.9008
breasttissue	0.7113	0.7005	0.6515	0.7455	0.6515	0.6944	0.5391	0.6663	0.5711
car	0.8279	0.6185	0.8449	0.8131	0.8524	0.6489	0.7898	0.7855	0.8273
contraception	0.5027	0.4773	0.5017	0.4739	0.5010	0.4699	0.4751	0.4964	0.4972
ecoli	0.8630	0.8547	0.8629	0.8510	0.8659	0.8576	0.7450	0.8456	0.8098
glass	0.6448	0.5813	0.6542	0.5746	0.6450	0.6342	0.4504	0.5988	0.6215
imageseg	0.9169	0.9103	0.9162	0.9210	0.9219	0.9088	0.7741	0.8157	0.8962
iris	0.9600	0.8893	0.9533	0.9467	0.9533	0.8813	0.7640	0.8320	0.9253
vehicle	0.8016	0.7978	0.7955	0.7872	0.7990	0.7941	0.6870	0.7550	0.7993
vertebral	0.8323	0.8458	0.8355	0.8484	0.8419	0.8439	0.7890	0.8432	0.8426
vowel	0.4762	0.4242	0.4957	0.3398	0.5065	0.4221	0.2273	0.3277	0.5017
wine	0.9776	0.9841	0.9608	0.9775	0.9719	0.9775	0.9843	0.9775	0.9717
yeast	0.5343	0.5841	0.5748	0.5175	0.5802	0.5818	0.4217	0.5590	0.5811

TABLE 2.6 – *Predictive ARI scores for each of the classification methods on all data sets. All numbers are out-of-sample ARI scores averaged over the five independent test folds. Maximum scores per dataset are determined on the full precision scores and are underlined.*

Dataset	GenSVM	LL C&S	DAG	OvA	OvO	C&S	LLW	MSVM^2	W&W
balancescale	0.8042	0.7355	0.8042	0.7238	0.8042	0.7466	0.6634	0.6653	0.7698
breasttissue	0.5222	0.4964	0.4591	0.5723	0.4787	0.4755	0.4043	0.5585	0.4655
car	0.5381	0.3290	0.5545	0.5238	0.5491	0.3131	0.4874	0.4808	0.5337
contraception	0.0762	0.0532	0.0757	0.0535	0.0747	0.0525	0.0393	0.0658	0.0699
ecoli	0.7668	0.7606	0.7755	0.7652	0.7776	0.7578	0.6236	0.7499	0.7385
glass	0.2853	0.2478	0.2970	0.2346	0.2910	0.2792	0.1776	0.2494	0.2861
imageseg	0.8318	0.8221	0.8280	0.8402	0.8390	0.8193	0.6422	0.6810	0.7991
iris	0.8783	0.7057	0.8609	0.8384	0.8609	0.6879	0.5549	0.6057	0.7918
vehicle	0.6162	0.6009	0.6057	0.5979	0.6057	0.5925	0.4933	0.5397	0.6121
vertebral	0.6649	0.6797	0.6606	0.6862	0.6778	0.6742	0.6480	0.6859	0.6836
vowel	0.2472	0.2474	0.2895	0.1624	0.3218	0.2257	0.1425	0.2043	0.2767
wine	0.9320	0.9585	0.8848	0.9378	0.9200	0.9362	0.9498	0.9332	0.9250
yeast	0.2519	0.2501	0.2415	0.2419	0.2477	0.2534	0.1301	0.2235	0.2501

TABLE 2.7 – *Computation time in seconds for each of the methods on all datasets. Values are averaged over the five nested CV splits. Minimum values per dataset are underlined. Note that the size of the grid search is 18 times larger in GenSVM than in other methods.*

Dataset	GenSVM	LL C&S	DAG	OvA	OvO	C&S	LLW	MSVM^2	W&W
balancescale	$\underline{44.3}$	88.0	86.4	155.8	84.9	34549	73671	79092	35663
breasttissue	136.0	52.9	$\underline{3.8}$	65.2	3.8	28782	74188	38625	81961
car	$\underline{251.2}$	1239.1	1513.0	4165.2	1517.4	47408	95197	46978	85050
contraception	$\underline{82.5}$	1128.5	1948.3	5079.1	1913.4	45163	88844	43402	40335
ecoli	603.0	88.9	$\underline{34.7}$	183.2	34.8	28907	95989	39590	131571
glass	254.6	110.8	48.6	198.2	$\underline{47.7}$	27938	89073	37194	108499
imageseg	558.2	67.1	$\underline{2.4}$	151	3.2	32691	73300	48576	97218
iris	55.7	13.8	1.9	32.9	$\underline{1.5}$	12822	47196	38060	77409
vehicle	$\underline{186.4}$	376.8	307.9	1373.3	309.9	37605	49988	40665	43511
vertebral	$\underline{23.5}$	66.6	24.4	63.1	24.3	24716	70798	36168	23888
vowel	1282.4	463.9	$\underline{83.7}$	3900.2	86.1	36270	95036	49924	82990
wine	129.6	$\underline{0.1}$	0.2	0.2	0.2	12854	70439	18389	41018
yeast	1643.3	$\underline{1181.7}$	1434.6	4251.1	1423.6	44112	103240	56603	86802

TABLE 2.8 – *Average computation time in seconds per hyperparameter configuration for each of the methods on all datasets. Values are averaged over the five nested CV splits. Minimum values per dataset are determined on the full precision values and are underlined.*

Dataset	GenSVM	LL C&S	DAG	OvA	OvO	C&S	LLW	MSVM^2	W&W
balancescale	$\underline{0.130}$	4.632	4.546	8.199	4.468	1818	3877	4163	1877
breasttissue	0.398	2.785	$\underline{0.201}$	3.434	0.201	1515	3905	2033	4314
car	$\underline{0.734}$	65.217	79.629	219.221	79.863	2495	5010	2473	4476
contraception	$\underline{0.241}$	59.396	102.544	267.319	100.704	2377	4676	2284	2123
ecoli	$\underline{1.763}$	4.680	1.828	9.643	1.831	1521	5052	2084	6925
glass	$\underline{0.744}$	5.832	2.559	10.432	2.511	1470	4688	1958	5710
imageseg	1.632	3.530	$\underline{0.128}$	7.947	0.167	1721	3858	2557	5117
iris	0.163	0.725	0.101	1.729	$\underline{0.081}$	675	2484	2003	4074
vehicle	$\underline{0.545}$	19.830	16.206	72.282	16.308	1979	2631	2140	2290
vertebral	$\underline{0.069}$	3.504	1.286	3.32	1.279	1301	3726	1904	1257
vowel	$\underline{3.750}$	24.415	4.406	205.274	4.533	1909	5002	2628	4368
wine	0.379	$\underline{0.003}$	0.010	0.011	0.010	677	3707	968	2159
yeast	$\underline{4.805}$	62.197	75.506	223.74	74.925	2322	5434	2979	4569

3

Fast Meta-Learning for Adaptive Hierarchical Classifier Design

G.J.J. van den Burg and A.O. Hero III

Abstract

We propose a new splitting criterion for a meta-learning approach to multiclass classifier design that adaptively splits the classes into a tree-structured hierarchy of increasingly difficult binary classification problems. The classification tree is constructed from empirical estimates of the Henze-Penrose bounds on the pairwise Bayes misclassification rates that rank the binary subproblems in terms of difficulty of classification. The proposed empirical estimates of the Bayes error rate are computed from the minimal spanning tree of the samples from each pair of classes. Moreover, a meta-learning technique is presented for quantifying the one-vs-rest Bayes error rate for each individual class from a single MST on the entire dataset. Extensive simulations on benchmark datasets show that the proposed hierarchical method can often be learned much faster than competing methods, while achieving competitive accuracy.

This chapter is based on Van den Burg and Hero (2017).

3.1 Introduction

The Bayes error rate (BER) is a central concept in the statistical theory of classification. It represents the error rate of the Bayes classifier, which assigns a label to an object corresponding to the class with the highest posterior probability. By definition, the Bayes error represents the smallest possible average error rate that can be achieved by any decision rule (Wald, 1947). Because of these properties, the BER is of great interest both for benchmarking classification algorithms as well as for the practical design of classification algorithms. For example, an accurate approximation of the BER can be used for classifier parameter selection, data dimensionality reduction, or variable selection. However, accurate BER approximation is difficult, especially in high dimension, and thus much attention has focused on tight and tractable BER bounds. This chapter proposes a model-free approach to designing multiclass classifiers using a bias-corrected BER bound estimated directly from the multiclass data.

There exists several useful bounds on the BER that are functions of the class-dependent feature distributions. These include information theoretic divergence measures such as the Chernoff α-divergence (Chernoff, 1952), the Bhattacharyya divergence (Kailath, 1967), or the Jensen-Shannon divergence (Lin, 1991). Alternatively, arbitrarily tight bounds on performance can be constructed using sinusoidal or hyperbolic approximations (Hashlamoun et al., 1994, Avi-Itzhak and Diep, 1996). These bounds are functions of the unknown class-dependent feature distributions.

Recently, Berisha et al. (2016) introduced a divergence measure belonging to the family of f-divergences which tightly bounds the Bayes error rate in the binary classification problem. The bounds on the BER obtained with this measure are tighter than bounds derived from the Bhattacharyya or Chernoff bounds. Moreover, this divergence measure can be estimated nonparametrically from the data without resorting to density estimates of the distribution functions. Inspired by the Friedman-Rafsky multivariate runs test (Friedman and Rafsky, 1979), estimation is based on computing the Euclidean minimal spanning tree (MST) of the data, which can be done in approximately $O(n \log n)$ time. In this chapter we propose improvements to this estimator for problems when there are unequal class priors and apply the improved estimator to the adaptive design of a hierarchical multiclass classifier. Furthermore, a fast method is proposed for bounding the Bayes error rate of individual classes which only requires computing a single minimal spanning tree over the entire set of samples. Thus our proposed method is faster than competing methods that use density plug-in estimation of divergence or observed misclassification rates of algorithms, such as SVM or logistic regression, which involve expensive parameter tuning.

Quantifying the complexity of a classification problem has been of significant interest (Ho and Basu, 2002) and it is clear that a fast and accurate estimate of this complexity has many practical applications. For instance, an accurate complexity estimator allows the researcher to assess a priori whether a given classification problem is *difficult* to classify or not. In a multiclass problem, a pair of classes which are difficult to disambiguate could potentially be merged or could be designated for additional data collection. Moreover, an accurate estimate of the BER could be used for variable selection, an application that was explored previously in Berisha et al. (2016). In Section 3.3 further applications of the BER estimates to multiclass classification are presented and evaluated.

There are many methods available for the design of multiclass classification algorithms, including: logistic regression (Cox, 1958); support vector machines (Cortes and Vapnik, 1995); and neural networks (McCulloch and Pitts, 1943). It is often the case that classifier performance will be better for some classes than for others, for instance due to sample imbalance in the training set. Often classifier designs apply weights to the different classes in order to reduce the effect of such imbalances on average classifier accuracy (Lu et al., 1998, Qiao and Liu, 2009). We take a different and more general approach that incorporates an empirical determination of the relative difficulties of classifying between different classes. Accurate empirical estimates of the BER are used for this purpose. A multiclass classifier is presented in Section 3.4 that uses MST-based BER estimates to create a hierarchy of binary subproblems that increase in difficulty as the algorithm progresses. This way, the classifier initially works on easily decidable subproblems before moving on to more difficult multiclass classification problems.

This chapter is organized as follows. The theory of the nonparametric Bayes error estimator of Berisha et al. (2016) will be reviewed in Section 3.2. We will introduce a bias correction for this estimator, motivate the use of the estimator for multiclass classification, and discuss computational complexity. Section 3.3 will introduce applications of the estimator to meta-learning in multiclass classification. A novel hierarchical classification method will be introduced and evaluated in Section 3.4. Section 3.5 provides concluding remarks.

3.2 An Improved BER Estimator

In this section the theory of the Henze-Penrose estimator of the Bayes error rate will be described. A central concept underlying this estimator is the minimal spanning tree. For the reader unfamiliar with this subject, a brief primer is given in the next section. Subsequently, the BER estimator will be described and an improvement of the estimator will be derived for the case where class prior probabilities are unequal. Next, the application of the estimator in multiclass classification problems is considered. Finally, computational considerations and robustness analyses are presented.

3.2.1 *Primer on Minimal Spanning Trees*

Let $G = (V, E, w)$ denote a *weighted* graph with a set of vertices V, a set of edges E, and a weight function $w : E \to \mathbb{R}$. Each edge $e \in E$ connects two vertices of the set V and can therefore be denoted by $e_{ij} = (v_i, v_j)$. It is assumed here that graphs are *undirected*, such that $e_{ij} = e_{ji}$. Furthermore, only graphs are considered where the set of edges is unique and where there are no loops that connect a vertex to itself. Finally, the weight function w assigns a weight to each edge $e \in E$. A common choice for the weight function is the Euclidean distance between the vertices, but other functions can also be used.

With this definition, a *complete* graph is a graph where each vertex in V is connected to each other vertex. By denoting the number of vertices with $n = |V|$ the number of edges in a complete graph can be written as $\frac{1}{2}n(n-1)$. In general, every graph with the same vertex set V as the complete graph but with a different edge set is called a *subgraph* of the complete graph. Two types of subgraphs can then be distinguished, namely *connected* and *unconnected* subgraphs. A connected subgraph is one in which there exists a path from each vertex to each other vertex. Among the connected subgraphs of the complete graphs, a further distinction can be made between graphs with and without so-called *cycles*. A cycle in a graph occurs when there is more than one path between a pair of vertices. Connected graphs without cycles are called *trees* and trees that connect all vertices in G are called *spanning* trees of G. The number of edges in a spanning tree thus equals $n - 1$.

The weight of a spanning tree can be computed as the sum of the weights of each of its edges. If the spanning tree is denoted by $T = (V, E', w)$ where E' is the set of edges in the spanning tree, the weight of the spanning tree can be computed using the weight function w as

$$W = \sum_{e \in E'} w(e). \tag{3.1}$$

A natural question is then to find the spanning tree of a graph with the smallest total weight W. This tree is called the *minimal* spanning tree. Classic algorithms for constructing the minimal spanning tree of a graph are the algorithms by Borůvka (1926), Prim (1957), and Kruskal (1956). In the following section, the minimal spanning tree of a dataset will play an important role.

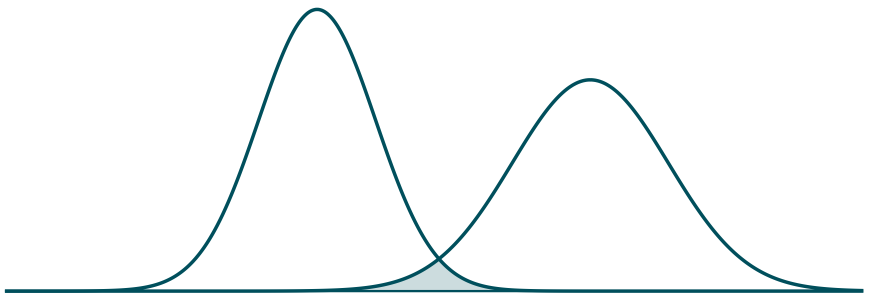


FIGURE 3.1 – *Illustration of the Bayes error rate for a binary classification problem where the density functions are given by univariate Gaussians. The BER corresponds to the shaded area.*

3.2.2 *Estimating the Bayes error rate*

Consider the binary classification problem on a dataset $\mathscr{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$, where $\mathbf{x}_i \in \mathscr{X} \subseteq \mathbb{R}^d$ and $y_i \in \{1,2\}$. Denote the multivariate density functions for the two classes by $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ and the prior probabilities by p_1 and $p_2 = 1 - p_1$, respectively. Following Fukunaga (1990) the Bayes error rate of this binary classification problem can be expressed as

$$P_e(f_1, f_2) = \int \min\{p_1 f_1(\mathbf{x}), p_2 f_2(\mathbf{x})\} \,\mathrm{d}\mathbf{x}. \tag{3.2}$$

The BER gives the minimum average probability of error that can be obtained by any classifier (Wald, 1947). Figure 3.1 illustrates the concept of the BER for a one-dimensional binary classification problem.

Recently, Berisha et al. (2016) derived a tight bound on the BER that can be estimated directly from the data without a parametric model for the density or density estimation. This bound is based on a divergence measure introduced by Berisha and Hero (2015), defined as

$$D_{HP}(f_1, f_2) = \frac{1}{4p_1p_2}\left[\int \frac{(p_1 f_1(\mathbf{x}) - p_2 f_2(\mathbf{x}))^2}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})}\,\mathrm{d}\mathbf{x} - (p_1 - p_2)^2\right], \tag{3.3}$$

and called the Henze-Penrose divergence, as it is motivated by an affinity measure defined by Henze and Penrose (1999). In Berisha and Hero (2015) it was shown that (3.3) is a proper f-divergence as defined by Csiszár (1975).

Estimation of the Henze-Penrose (HP) divergence is based on the multivariate runs test proposed by Friedman and Rafsky (1979) and convergence of this test was studied by Henze and Penrose (1999). Let $\mathbf{X}_k = \left\{\mathbf{x}_i \in \mathscr{X} \subseteq \mathbb{R}^d : y_i = k\right\}$ with $k \in \{1,2\}$ denote the multidimensional features from two classed. Defined the class sample sizes $n_k = |\mathbf{X}_k|$, $k \in \{1,2\}$. Let the combined sample be denoted by $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$ where $n = |\mathbf{X}| = n_1 + n_2$ is the total number of samples from both classes. Define the complete weighted graph G over $\mathbf{X}$ as the graph connecting all n nodes $\{\mathbf{x}_i\}_{i=1}^n$ with edge weights $|e_{ij}| = \|\mathbf{x}_i - \mathbf{x}_j\|$ equal to Euclidean distances. The Euclidean minimal spanning tree that spans $\mathbf{X}$, denoted by T, is defined as the subgraph of G that is both connected and whose sum of edge weights is the smallest possible. The Friedman-Rafsky test statistic equals the number of edges in T that connect an observation from class 1 to an observation from class 2 and is denoted by $R_{1,2} = R(\mathbf{X}_1, \mathbf{X}_2) = \left|\{e_{ij} = (\mathbf{x}_i, \mathbf{x}_j) \in T : \mathbf{x}_i \in \mathbf{X}_1, \mathbf{x}_j \in \mathbf{X}_2\}\right|$.

Henze and Penrose (1999) show that if $n_1 \to \infty$ and $n_2 \to \infty$ in a linked manner such that $n_1/(n_1+n_2) \to \delta \in (0,1)$ then,

$$\frac{R_{1,2}}{n_1+n_2} \to 2p_1p_2 \int \frac{f_1(\mathbf{x})f_2(\mathbf{x})}{p_1f_1(\mathbf{x})+p_2f_2(\mathbf{x})}\,\mathrm{d}\mathbf{x} \qquad \textit{almost surely.}^1 \tag{3.4}$$

Based on the above result, Berisha et al. (2016) show that under the same conditions

$$1 - \frac{nR_{1,2}}{2n_1n_2} \to D_{HP}(f_1,f_2) \qquad \textit{almost surely.} \tag{3.5}$$

Thus, the number of cross connections between the classes in the Euclidean MST is inversely proportional to the divergence between the respective probability density functions of these classes. Figure 3.2 illustrates this result for a hypothetical dataset with two-dimensional Gaussian distributions. As can be seen, distributions that have a larger overlap – and therefore a lower divergence – have a larger number of cross connections in the MST.

Finally, the HP-divergence can be used to bound the Bayes error rate, $P_e(f_1,f_2)$, following Theorem 2 of Berisha et al. (2016)

$$\tfrac{1}{2} - \tfrac{1}{2}\sqrt{u_{HP}(f_1,f_2)} \le P_e(f_1,f_2) \le \tfrac{1}{2} - \tfrac{1}{2}u_{HP}(f_1,f_2), \tag{3.6}$$

where

$$u_{HP}(f_1,f_2) = 4p_1p_2D_{HP}(f_1,f_2) + (p_1-p_2)^2. \tag{3.7}$$

Since $u_{HP}(f_1,f_2)$ is a function of $D_{HP}(f_1,f_2)$, the above convergence result (3.5) can be extended to $u_{HP}(f_1,f_2)$ to find

$$1 - \frac{2R_{1,2}}{n_1+n_2} \to u_{HP}(f_1,f_2) \tag{3.8}$$

[1]Almost sure convergence implies that a sequence of random variables X_n converges to X with probability 1, i.e. $\Pr[\lim_{n\to\infty} X_n = X] = 1$.

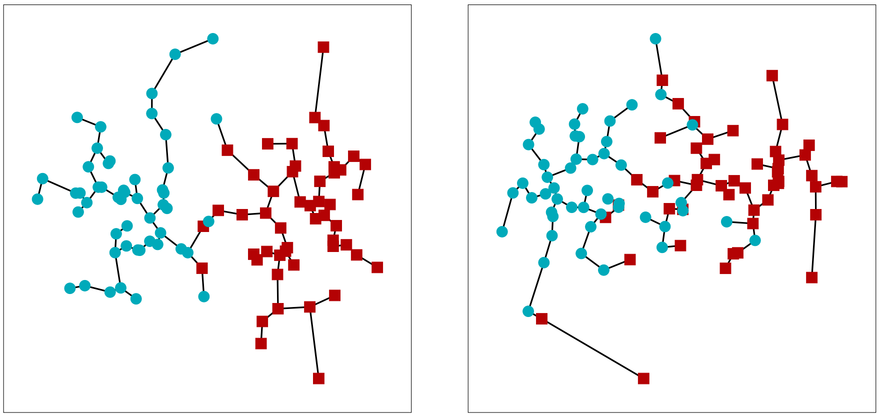

(a) large separation (b) small separation

FIGURE 3.2 – *Illustration of the relationship between the number of cross connections in the MST and the Henze-Penrose divergence measure. In the large separation case (a) the number of cross connections is smaller than in the small separation case (b). This corresponds respectively with a small and a large divergence between the class distributions. Adapted from Berisha et al. (2016).*

with the same convergence criteria. Averaging these bounds yields an estimate of the BER given by

$$\hat{P}_e(f_1, f_2) = \tfrac{1}{2} - \tfrac{1}{4}\sqrt{u_{HP}(f_1, f_2)} - \tfrac{1}{4} u_{HP}(f_1, f_2). \tag{3.9}$$

In the following, this estimator will be referred to as the HP-estimator of the BER.

3.2.3 *A modified HP-estimator for unequal class priors*

It is important to consider the performance of the HP-estimator in practice. Due to the construction of the Friedman-Rafsky statistic $R_{1,2}$ it is possible that in the finite sample case a value of $R_{1,2}$ is obtained which exceeds $n/2$. Because this would give a negative estimate for u_{HP} the following correction is typically applied in practice,

$$R'_{1,2} = \min\{n/2, R_{1,2}\}. \tag{3.10}$$

Furthermore, since there will always be at least one cross-connection in the MST, $R_{1,2}$ should be reduced by 1 to ensure that in the perfectly separable case $D_{HP} = 1$ as expected.

To illustrate the performance of the HP-estimator in practice, consider a binary classification problem with $n = n_1 + n_2 = 1000$ samples. The samples are drawn from two independent bivariate Gaussian distributions with equal covariance matrices for which the density functions are known. For this scenario the true BER can be computed exactly as

$$P_e(f_1,f_2) = \frac{1}{2} - \frac{p_1}{2}\operatorname{erf}\left(\frac{\log(p_1/p_2) + \frac{1}{2}\eta}{\sqrt{2\eta}}\right) + \frac{p_2}{2}\operatorname{erf}\left(\frac{\log(p_1/p_2) - \frac{1}{2}\eta}{\sqrt{2\eta}}\right), \tag{3.11}$$

with

$$\eta = \left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right) \tag{3.12}$$

following Fukunaga (1990). In addition to the true BER and the HP-estimator, it is also interesting to consider the performance of the commonly used Bhattacharyya bound of the BER (Bhattacharyya, 1946, Kailath, 1967), given by

$$P_{BC} = \tfrac{1}{4} + \tfrac{1}{2}\sqrt{p_1 p_2}\rho - \tfrac{1}{4}\sqrt{1 - 4p_1 p_2 \rho^2} \tag{3.13}$$

where ρ is the Bhattacharyya coefficient given by

$$\rho(f_1,f_2) = 2\int \sqrt{p_1 p_2 f_1(\mathbf{x}) f_2(\mathbf{x})}\,\mathrm{d}\mathbf{x}. \tag{3.14}$$

For multivariate Gaussians ρ can be computed exactly using the true underlying density functions (Kailath, 1967). This illustrates the problem of using the Bhattacharyya estimate of the BER in practice, because the exact underlying densities are typically unknown. Figure 3.3(a) shows that the HP-estimator is closer to the true BER than the Bhattacharyya bound. This result was illustrated by Berisha et al. (2016) for $p_1 = p_2$ and is confirmed here for $p_1 \neq p_2$.

However, Figure 3.3(a) also shows that a significant bias occurs in the HP-estimate of the BER when the distance between classes is small. This bias can be corrected in a similar fashion as the correction introduced above in (3.10). Considering that $P_e = \min\{p_1, p_2\}$ if $f_1 \to f_2$, the solution of the equation $\hat{P}_e = \min\{\hat{p}_1, \hat{p}_2\}$ for $R_{1,2}$ gives the correction for the bias as

$$R''_{1,2} = \min\{\gamma, R_{1,2}\}, \tag{3.15}$$

with

$$\gamma = 2n\min\{\hat{p}_1, \hat{p}_2\} - \tfrac{3}{4}n + \tfrac{1}{4}n\sqrt{9 - 16\min\{\hat{p}_1, \hat{p}_2\}}. \tag{3.16}$$

where $\hat{p}_1 = n_1/n$ and $\hat{p}_2 = n_2/n$ are estimates of the true prior probabilities. Figure 3.3(b) shows the effect of this bias correction on the accuracy of the HP-

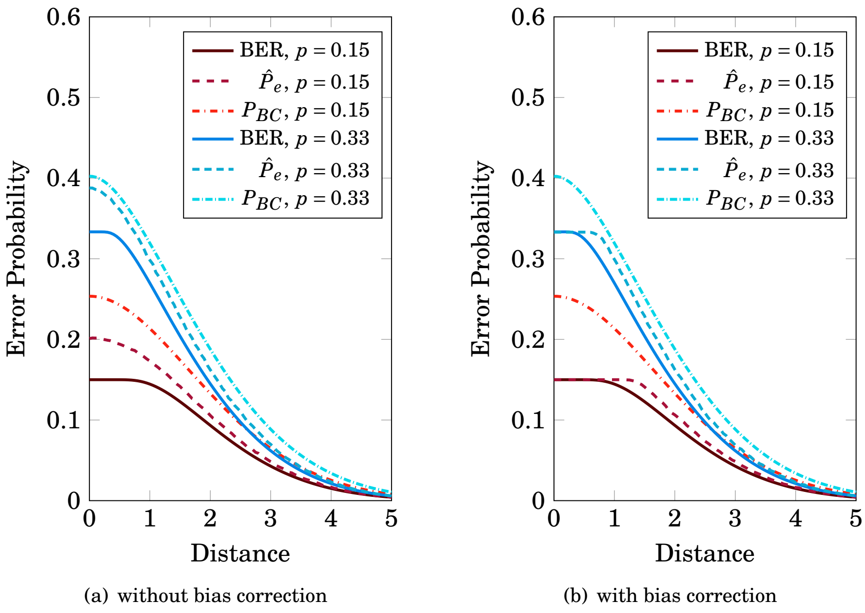


FIGURE 3.3 – *Illustration of the estimates of the Bayes error rate for two different values of the prior probability p and for spherical bivariate Gaussian distributions with increasing separation between the classes as measured by the Euclidean distance between the class means,* $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|$. *The true BER, the Bhattacharyya bound (P_{BC}) requiring the true class distributions, and the HP-estimator ($\hat{P}_e$) based on an empirical sample are shown. Figures (a) and (b) show the estimates for* $p = 0.15$ *and* $p = 1/3$, *respectively with and without the bias correction on the HP-estimator. The HP-estimator was implemented using* $n = 1000$ *samples* ($n_1 = pn$ *and* $n_2 = n - n_1$) *and averaged over 200 trials.*

estimator. As can be seen, the bias correction significantly improves the accuracy of the HP-estimator for when the class distributions are not well separated.

3.2.4 *Multiclass classification*

Here the HP-estimator is applied to multiclass classification problems by extending the bias corrected HP-estimator to a multiclass Bayes error rate. The original multiclass HP-estimator has been defined by Wisler et al. (2016) and we show how the framework can be applied to hierarchical multiclassifier design.

Consider a multiclass problem with K classes with $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y_i \in \{1,\dots,K\}$, with prior probabilities p_k and density functions $f_k(\mathbf{x})$ for $k = 1,\dots,K$ such that $\sum_k p_k = 1$. Then, the BER can be estimated for each pair of classes using the bias-corrected HP-estimator $\hat{P}_e(f_k, f_j)$ presented in (3.15). The binary classification problem with the largest BER estimate is defined as most difficult.

Recall that the largest BER that can be achieved in a binary classification problem with unequal class priors is equal to the value of the smallest prior probability. This makes it difficult to compare empirical estimates of the BER when class sizes are imbalanced. For example, in a classification problem with three classes where $p_1 = 0.2$ and $p_2 = p_3 = 0.4$, the binary problems involving class 1 will always have a smaller maximum probability of error, simply because it has a smaller prior probability. To correct for this, the HP-estimator for pairwise classification BERs can be normalized for class sizes using

$$\hat{P}'_e(f_k, f_l) = \frac{\hat{P}_e(f_k, f_l)}{\min\{\hat{p}_k, \hat{p}_l\}}. \tag{3.17}$$

This normalization places the HP-estimate in the interval $[0,1]$ and makes it possible to more accurately compare the BER estimates of different binary problems.

In practice it can also be of interest to understand how difficult it is to discriminate each individual class. By reducing the multiclass problem to a One-vs-Rest classification problem, it is straightforward to define a *confusion rate* for a given class k. This represents the fraction of instances that are erroneously assigned to class k and the fraction of instances which are truly from class k that are assigned to a different class. Formally, define the confusion rate for class k as

$$C_k(y, \hat{y}) = \frac{|\{i : \hat{y}_i = k, y_i \neq k\}| + |\{i : \hat{y}_i \neq k, y_i = k\}|}{n}, \tag{3.18}$$

with $\hat{y}_i$ the predicted class for instance i. Recall that the Bayes error rate is the error rate of the Bayes classifier, which assigns an instance $\mathbf{x}$ to class $k = \arg\max_l p_l f_l(\mathbf{x})$. Hence, the BER for a single class k equals the error of assigning to a class $l \neq k$ when the true class is k and the total error of assigning to class k when the true class is $c \neq k$, thus

$$P_{e,k} = \int\limits_{\max\limits_{l\neq k}\{p_l f_l(\mathbf{x})\} \geq p_k f_k(\mathbf{x})} p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} + \sum_{c\neq k} \int\limits_{\max\limits_{l\neq k}\{p_l f_l(\mathbf{x})\} < p_k f_k(\mathbf{x})} p_c f_c(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{3.19}$$

Figure 3.4 illustrates this error for a one-dimensional multiclass classification problem.

We make two observations about this *One-vs-Rest* Bayes error rate (OvR-BER). First, the OvR-BER for class k is smaller than the sum of the binary BERs for the problems involving class k (see Appendix 3.A). Second, the OvR-BER

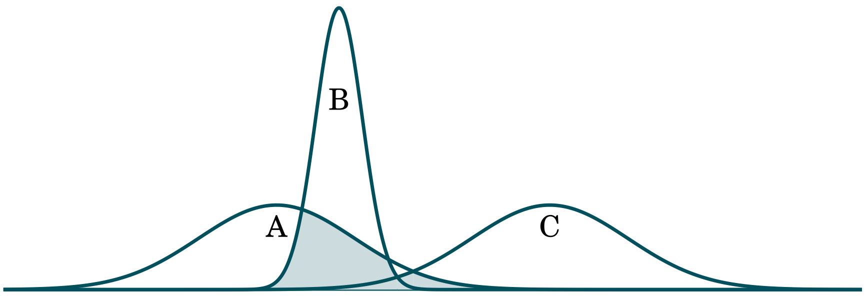


FIGURE 3.4 – *Illustration of the One-vs-Rest Bayes error rate for a multiclass classification problem with* 3 *classes. The OvR-BER of class A corresponds to the shaded area.*

can be estimated using the Henze-Penrose divergence with $R(\mathbf{X}_k, \bigcup_{l\neq k}\mathbf{X}_l)$, which yields the estimate $\hat{P}_{e,k}$. A computational advantage of using the OvR-BER in multiclass problems is that the MST only has to be computed only once on the set $\mathbf{X}$, since the union of $\mathbf{X}_k$ and $\cup_{l\neq k}\mathbf{X}_l$ is equal to $\mathbf{X}$. Therefore, $R(\mathbf{X}_k, \bigcup_{l\neq k}\mathbf{X}_l)$ can be computed for all k from the single MST on $\mathbf{X}$ by keeping track of the labels of each instance.

3.2.5 *Computational Considerations*

The construction of the minimal spanning tree lies at the heart of the HP-estimator of the BER, so it is important to use a fast algorithm for the MST construction. Since the HP-estimator is based on the Euclidean MST the dual-tree algorithm by March et al. (2010) can be applied. This algorithm is based on the construction of Borůvka (1926) and implements the Euclidean MST in approximately $O(n \log n)$ time. For larger datasets it can be beneficial to partition the space into hypercubes and construct the MST in each partition.

A simple way to improve the robustness of the HP-estimator is to use multiple orthogonal MSTs and average the number of cross-connections (Friedman and Rafsky, 1979). Orthogonal MSTs have no common edges between them: the first MST is constructed on all edges, the second is constructed on all edges except those in the first MST, the third is constructed on all edges except those in the previous two MSTs, etc. Computing orthogonal MSTs is not straightforward in the dual-tree algorithm of March et al. (2010), but is easy to implement in MST algorithms that use a pairwise distance matrix such as that of Whitney (1972). Figure 3.5 shows the empirical variance of the HP-estimator for different numbers of orthogonal MSTs as a function of the separation between the classes. As expected, the variance decreases as the number of orthogonal MSTs increases,

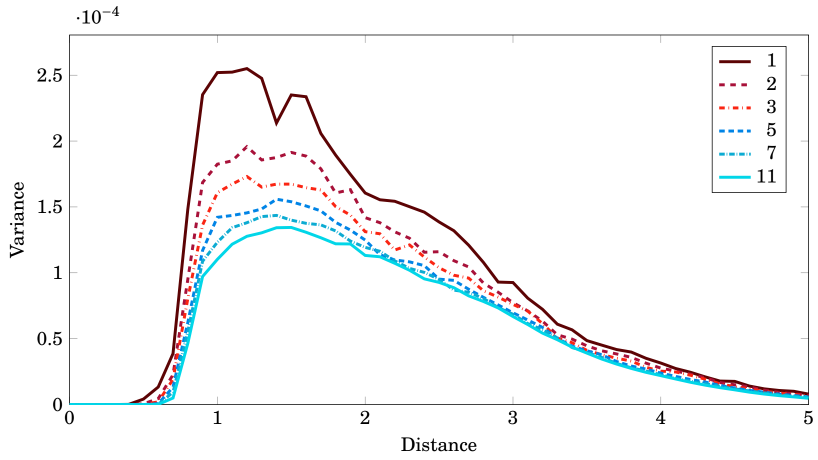
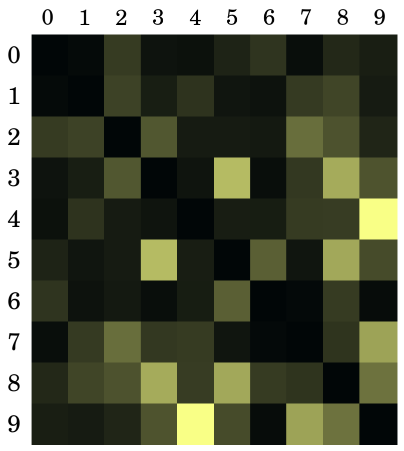


FIGURE 3.5 – *Variance of the HP-estimator of the BER for varying number of orthogonal MSTs as a function of the separation between the classes. Results are based on 500 repetitions of a binary classification problem with* $n = 1000$, $d = 2$, $p_1 = 0.33$, *where class distributions are bivariate spherical Gaussians with different means.*

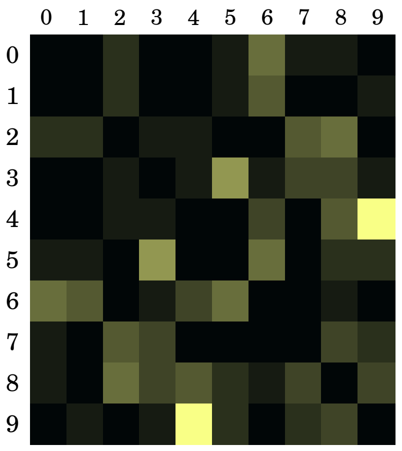
although the benefit of including more orthogonal MSTs also decreases when adding more MSTs. Therefore, 3 orthogonal MSTs are typically used in practice.

3.3 Meta-Learning of optimal classifier accuracy

Applying the HP-estimator to meta-learning problems creates a number of opportunities to assess the difficulty of a classification problem before training a classifier. For example, given a multiclass classification problem it may be useful to know which classes are difficult to distinguish from each other and which classes are easy to distinguish. Figure 3.6(a) shows an illustration of this for handwritten digits in the well-known MNIST dataset (LeCun et al., 1998). This figure shows a heat map where each square corresponds to an estimate of the BER for a binary problem in the training set. From this figure it can be seen that the digits 4 and 9 are difficult to distinguish, as well as the digits 3 and 5. This information can be useful for the design of a classifier, to ensure for instance that higher weights are placed on misclassifications of the more difficult number pairs if correct classification of these pairs is of importance to the end-task. In Figure 3.6(b) a similar heat map is shown based on misclassified instances of LeNet-5 (LeCun et al., 1998) on the test set. This figure shows the symmetric confusion matrix based on the 82 misclassified instances. As can be seen, this



(a)



(b)

FIGURE 3.6 – *Heat maps illustrating the difficulty of distinguishing between different handwritten digits in the MNIST dataset of LeCun et al. (1998). Brighter squares correspond to higher values. In (a) the heat map of the BER estimates on the training data is shown. In (b) the heat map is shown of the 82 misclassifications made by LeNet-5 on the test data (LeCun et al., 1998). From (a) it can be readily identified that the numbers 3 and 5 are difficult to distinguish, as well as 4 and 9. Easy to distinguish number pairs are for instance 6 and 7, and 0 and 1. This pattern is reflected in the results on the test data shown in (b).*

figure closely corresponds to the heat map on the training data, which confirms the predictive accuracy of the HP-estimator for real data.

Another example of the accuracy of the BER estimates for multiclass classification problems is given in Figure 3.7. In this figure, OvR-BER estimates $\hat{P}_{e,k}$ and class confusion rates $C_k(y, \hat{y})$ are shown for the Chess dataset ($n = 28056$, $d = 34$, $K = 18$) obtained from the UCI repository (Bache and Lichman, 2013). This dataset was split into a training dataset (70%) and a test dataset (30%) and the OvR-BER estimates were computed on the training dataset. These estimates are compared with the class error rates obtained from out-of-sample predictions of the test dataset using GenSVM (Van den Burg and Groenen, 2016). This figure shows that the OvR-BER estimates are accurate predictors of classification performance. The classes that are relatively difficult to classify may benefit from increasing misclassification weights.

The BER estimates can also be applied to feature selection and, in particular, to the identification of useful feature transformations of the data. A feature selection strategy based on forward selection was outlined in Berisha et al. (2016).

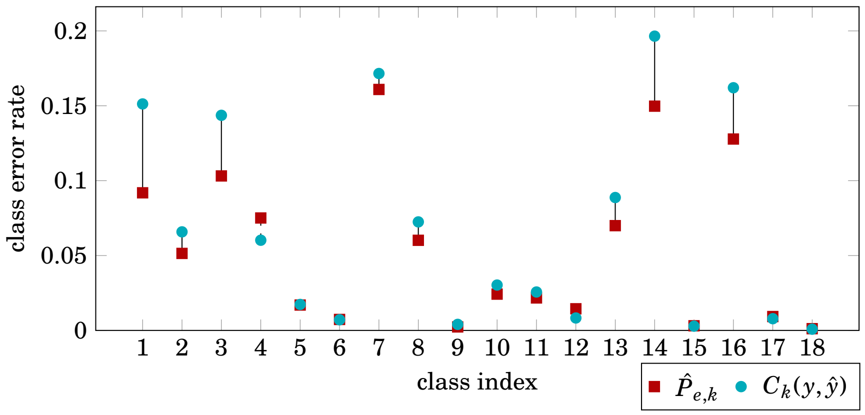


FIGURE 3.7 – *OvR-BER estimates and test set error rates for each class in the Chess dataset. It can be seen that for most classes the OvR-BER estimate is an accurate predictor for test set performance. For classes where there is a difference between the BER and the test set error the BER generally gives a lower bound on the test set performance. Test set results were obtained with the GenSVM classifier (Van den Burg and Groenen, 2016).*

At each feature selection stage, this algorithm adds the feature which gives the smallest increase in the BER estimate. Berisha et al. (2016) show that this feature selection strategy quickly yields a subset of useful features for the classification problem.

Because the BER estimate is a fast and asymptotically consistent estimate of a bound on classification performance, it is easy to try a number of potential feature transformations and use the one with the smallest BER estimate in the classifier. This can be useful both for traditional feature transformations such as PCA (Pearson, 1901) and Laplacian Eigenmaps (Belkin and Niyogi, 2003), but also for commonly used kernel transformations in SVMs. For a researcher this can significantly reduce the time needed to train a classifier on different transformations of the data. In a multiclass setting where the One-vs-One strategy is used, one can even consider a different feature transformation for each binary subproblem. When using a unified classification method one can consider feature transformations which reduce the average BER estimate or the worst-case BER estimate.

Note that a feature transformation which reduces the dimensionality of the dataset without increasing the BER estimate can be considered beneficial, as many classification methods are faster for low-dimensional datasets. For instance, applying PCA with 2 components on the Chess dataset only slightly increases the BER estimates for two classes, while remaining the same for the other classes.

Thus, a classifier will likely achieve comparable accuracy with this transformed dataset, but will be much faster to train since the dimensionality can be reduced from $d = 34$ to $d = 2$.

3.4 Hierarchical Multiclass Classification

In this section a novel hierarchical multiclass SVM classifier is introduced which is based on uncertainty clustering. The BER estimate can be considered a measure of the irreducible uncertainty of a classification problem, as a high BER indicates an intrinsically difficult problem. This can be used to construct a tree of binary classification problems that increase in difficulty along the depth of the tree. By fitting a binary classifier (such as an SVM) at each internal node of the tree, a classification method is obtained which proceeds from the easier binary subproblems to the more difficult binary problems.

Similar divide-and-conquer algorithms have been proposed (Schwenker and Palm, 2001, Takahashi and Abe, 2002, Frank and Kramer, 2004, Vural and Dy, 2004, Tibshirani and Hastie, 2007, among others). See Lorena et al. (2008) for a review. These approaches often apply a clustering method to create a grouping of the dataset into two clusters, repeating this process recursively to form a binary tree of classification problems. In Lorena and De Carvalho (2010) several empirical distance measures are used as indicators of separation difficulty between classes, which are applied in a bottom-up procedure to construct a classification tree. Finally, in El-Yaniv and Etzion-Rosenberg (2010) the Jensen-Shannon divergence is used to bound the BER with inequalities from Lin (1991) and a classification tree is constructed using a randomized heuristic procedure. Unfortunately, the Jensen-Shannon divergence implementation requires parametric estimation of distribution functions. Moreover, for the equiprobable case the upper bound on the BER obtained with the Jensen-Shannon divergence can be shown to be less tight than that obtained with the HP-divergence (see Appendix 3.B). Because of this, these estimates of the BER may be less accurate than those obtained with the proposed HP-estimator.

To construct the hierarchical classification tree a complete weighted graph $G = (V, E, w)$ is created where the vertices correspond to the classes and the weight of the edges equals the HP-estimate for that binary problem. Formally, let $V = \{1, \ldots, K\}$, $E = \{(i, j) : i, j \in V, i \neq j\}$ and define the edge weight $w(e)$ for $e \in E$ as $w(e) = w(i, j) = \hat{P}'_e(f_i, f_j)$. In the HP-estimator $\hat{P}'_e(f_i, f_j)$ the bias correction (3.15) and the normalization (3.17) are used. By recursively applying min-cuts to this graph a tree of binary classification problems is obtained which increase in difficulty along the depth of the tree. Min-cuts on this weighted graph can be

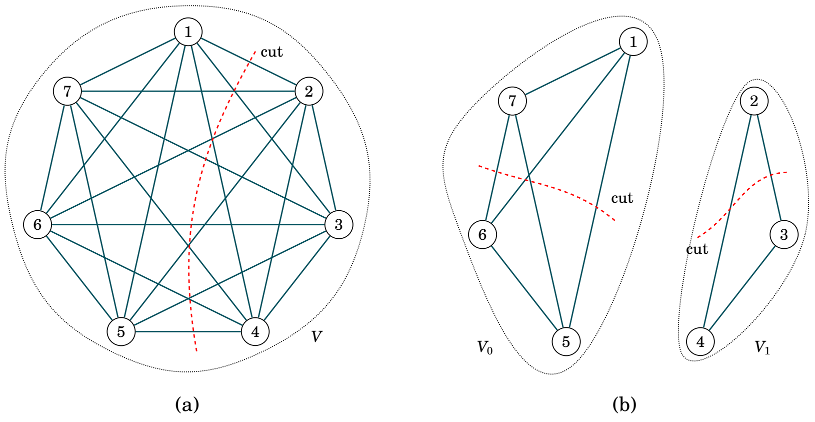


FIGURE 3.8 – *Illustration of the min-cut procedure for a multiclass classification problem with $K = 7$ classes. The first cut in (a) splits the vertices in the set V into two sets, V_0 and V_1, which are used in a binary classifier. Figure (b) illustrates the next step where both V_0 and V_1 are split further by cuts in the complete graph for each subset.*

computed using for instance the method of Stoer and Wagner (1997). Figure 3.8 illustrates this process for a multiclass classification problem with $K = 7$.

The tree construction can be described formally as follows. Starting with the complete weighted graph with vertices V, apply a min-cut algorithm to obtain the disjoint vertex sets V_0 and V_1 such that $V_0 \cup V_1 = V$. This pair of vertex sets then forms a binary classification problem with datasets $\mathbf{X}_0 = \{\mathbf{x}_i \in \mathbf{X} : y_i \in V_0\}$ and $\mathbf{X}_1 = \{\mathbf{x}_i \in \mathbf{X} : y_i \in V_1\}$. Recursively applying this procedure to the sets V_0 and V_1 until no further splits are possible yields a tree of binary classification problems, as illustrated in Figure 3.9.

In the remainder of this section the results of an extensive simulation study are presented, which aims to evaluate the performance of this hierarchical classifier on multiclass classification problems. The classifier that will be used in each binary problem in the tree will be a linear support vector machine, but in practice any binary classifier could be used in the algorithm. The implementation of the hierarchical classifier based on the linear binary SVM will be called *SmartSVM*.[2]

The experimental setup is comparable to that used in Van den Burg and Groenen (2016), where a nested cross-validation (CV) approach is used to reduce bias in classifier performance (Stone, 1974). From each original dataset five independent training and test datasets are generated. Subsequently, each

[2]The SmartSVM classifier and the meta-learning and BER estimation techniques presented in the previous sections have been implemented in the `smartsvm` Python package.

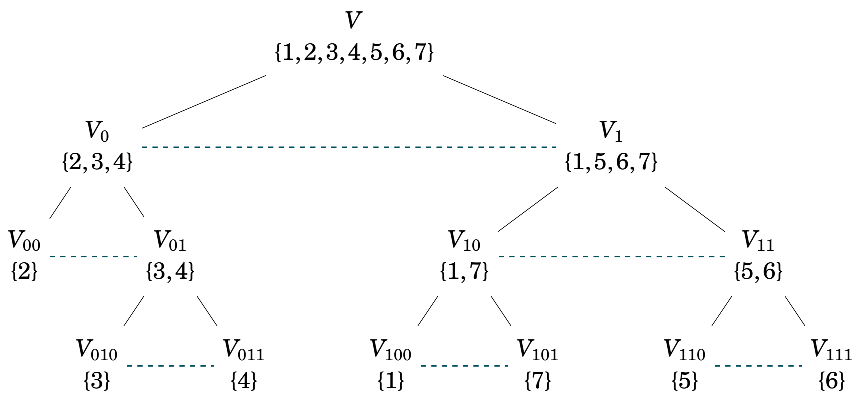


FIGURE 3.9 – *Illustration of the tree induced by the graph cutting procedure illustrated in Figure 3.8, for a hypothetical problem with $K = 7$ classes. Each dashed line in the tree indicates a point where a binary classifier will be trained.*

classification method is trained using 10 fold CV on each of the training datasets. Finally, the model is retrained on the entire training dataset using the optimal hyperparameters and this model is used to predict the test set. In the experiments 16 datasets are used of varying dimensions from which 80 independent test sets are constructed. The train and test datasets were generated using a stratified split, such that the proportions of the classes correspond to those in the full dataset. Table 3.1 shows the descriptive statistics of each of the datasets used. Datasets are collected from the UCI repository (Bache and Lichman, 2013) and the KEEL repository (Alcalá et al., 2010).

SmartSVM will be compared to five other linear multiclass SVMs in these experiments. Three of these alternatives are heuristic methods which use the binary SVM as underlying classifier, while two others are single-machine multiclass SVMs. One of the most commonly used heuristic approaches to multiclass SVMs is the One vs. One (OvO) method (Kreßel, 1999) which solves a binary SVM for each of the $K(K-1)$ pairs of classes. An alternative is the One vs. Rest (OvR) method (Vapnik, 1998) in which a binary SVM is solved for each of the $K-1$ binary problems obtained by separating a single class from the others. The directed acyclic graph (DAG) SVM was proposed by Platt et al. (2000) as an extension of the OvO approach. It has a similar training procedure as OvO, but uses a different prediction strategy. In the OvO method a voting scheme is used where the class with the most votes from each binary classifier becomes the predicted label. In contrast, the DAGSVM method uses a voting scheme where the least likely class is voted away until only one remains. Finally, two

TABLE 3.1 – *Dataset summary statistics for the datasets used in the experimental study. The final two columns denote the size of the smallest and the largest class, respectively. Datasets marked with an asterisk are collected from the KEEL dataset repository, all others are from the UCI repository.*

Dataset	Instances	Features	Classes	min n_k	max n_k
abalone	4177	8	20	14	689
fars*	100959	338	7	299	42116
flare	1066	19	6	43	331
kr-vs-k	28056	34	18	27	4553
letter	20000	16	26	734	813
nursery	12960	19	4	330	4320
optdigits	5620	64	10	554	572
pageblocks	5473	10	5	28	4913
pendigits	10992	16	10	1055	1144
satimage	6435	36	6	626	1533
segment	2310	19	7	330	330
shuttle	58000	9	6	23	45586
texture*	5500	40	11	500	500
wine red	1599	12	5	18	681
wine white	4898	12	6	20	2198
yeast	1479	8	9	20	463

single-machine multiclass SVMs are also compared: the method by Crammer and Singer (2002a) and GenSVM (Van den Burg and Groenen, 2016).

All methods are implemented in either `C` or `C++`, to ensure that speed of the methods can be accurately compared. The methods that use a binary SVM internally are implemented with LibLinear (Fan et al., 2008). LibLinear also implements a fast solver for the method by Crammer and Singer (2002a) using the algorithm proposed by Keerthi et al. (2008). For SmartSVM the Bayes error rates and the corresponding classification tree were calculated once for each training dataset as a preprocessing step. For most datasets the BERs were computed based on three orthogonal MSTs using the algorithm of Whitney (1972). For the two largest datasets (fars and shuttle) the BER was computed based on a single MST using the algorithm of March et al. (2010). Computing these MSTs was done in parallel using at most 10 cores. In the results on training time presented below the training time of SmartSVM is augmented with the preprocessing time.

The binary SVM has a cost parameter for the regularization term, which is optimized using cross validation. The range considered for this parameter is $C \in \{2^{-18}, 2^{-16}, \dots, 2^{18}\}$. The GenSVM method has additional hyperparameters which were varied in the same way as in the experiments of Van den Burg and

TABLE 3.2 – *Total training time per dataset in seconds, averaged over the five nested CV folds. Minimal values per dataset are underlined. As can be seen, SmartSVM is the fastest method on 10 out of 16 datasets.*

Dataset	C & S	DAG	GenSVM	OvO	OvR	SmartSVM
abalone	13332	146.5	86787	134.3	214.4	75.0
fars	158242	3108	205540	2866	5630	3158
flare	467.5	7.0	501.7	6.5	8.2	5.0
krkopt	86000	728.4	56554	680.4	1388	664.0
letter	31684	407.3	44960	381.5	1805	652.1
nursery	2267	74.7	1363	68.7	94.9	56.8
optdigits	88.0	20.0	31615	18.5	24.3	10.9
pageblocks	523.4	26.8	2001	25.2	48.6	27.6
pendigits	2499	30.2	6591	28.5	151.3	53.7
satimage	5184	74.7	2563	70.0	188.4	74.5
segment	377.1	7.1	1542	7.0	18.7	5.4
shuttle	7658	570.8	14618	561.9	1996	832.0
texture	588.1	34.5	14774	32.8	69.6	19.3
winered	706.4	12.3	542.7	11.1	18.3	8.8
winewhite	2570	59.2	2570	55.1	70.9	36.7
yeast	1209	16.7	1138	15.3	24.2	13.2

Groenen (2016). All experiments were performed on the Dutch National LISA Compute Cluster using the abed utility.[3]

The experiments are compared on training time and out-of-sample predictive performance. Table 3.2 shows the results for training time, averaged over the five nested cross validation folds for each dataset. As can be seen SmartSVM is the fastest method on 10 out of 16 datasets. This can be attributed to the smaller number of binary problems that SmartSVM needs to solve compared to OvO and the fact that the binary problems are smaller than those solved by OvR. The OvO method is the fastest classification method on the remaining 6 datasets. The single-machine multiclass SVMs by Crammer and Singer (2002a) and Van den Burg and Groenen (2016) both have larger computation times than the heuristic methods. Since GenSVM has a larger number of hyperparameters, it is interesting to look at the average time per hyperparameter configuration as well. In this case, GenSVM is on average faster than Crammer and Singer (2002a) due to the use of warm starts (see Appendix 3.C for additional simulation results).

Classification performance of the methods is reported using the adjusted Rand index (ARI) which corrects for chance (Hubert and Arabie, 1985). Use of this index as a classification metric has been proposed previously by Santos and

[3]See https://github.com/GjjvdBurg/abed.

TABLE 3.3 – *Out-of-sample predictive performance as measured by the adjusted Rand index. Although SmartSVM doesn't often achieve the maximum performance, there are several datasets for which SmartSVM outperforms One vs. One, or differs from the maximum performance in only the second or third decimal.*

Dataset	C & S	DAG	GenSVM	OvO	OvR	SmartSVM
abalone	0.0788	0.0898	0.0895	0.0898	0.0791	0.0595
fars	0.8127	0.8143	0.8085	0.8146	0.8134	0.8115
flare	0.4274	0.6480	0.6687	0.6496	0.4084	0.6544
krkopt	0.1300	0.1988	0.1779	0.2022	0.1512	0.1585
letter	0.6173	0.6991	0.5909	0.7118	0.5155	0.4533
nursery	0.8279	0.8175	0.8303	0.8157	0.8095	0.8138
optdigits	0.9721	0.9854	0.9732	0.9850	0.9686	0.9640
pageblocks	0.7681	0.7335	0.6847	0.7353	0.6696	0.7214
pendigits	0.9068	0.9566	0.8971	0.9597	0.8607	0.8817
satimage	0.7420	0.7652	0.7403	0.7672	0.7107	0.7607
segment	0.9013	0.9000	0.8812	0.8982	0.8354	0.8986
shuttle	0.9275	0.8543	0.8887	0.8543	0.6925	0.8038
texture	0.9936	0.9950	0.9888	0.9947	0.9865	0.8977
winered	1.0000	1.0000	0.9985	1.0000	0.8369	1.0000
winewhite	1.0000	1.0000	0.9998	1.0000	0.8131	1.0000
yeast	0.2595	0.2540	0.2521	0.2587	0.2433	0.2088

Embrechts (2009). Table 3.3 shows the predictive performance as measured with the ARI. As can be seen, SmartSVM obtains the maximum performance on two of the sixteen datasets. However, SmartSVM outperforms One vs. One on 3 datasets and outperforms One vs. Rest on 10 out of 16 datasets. The OvO and OvR methods are often used as default heuristic approaches for multiclass SVMs and are respectively the default strategies in the popular LibSVM (Chang and Lin, 2011) and LibLinear (Fan et al., 2008) libraries. Since SmartSVM is often faster than these methods, our results indicate a clear practical benefit to using SmartSVM for multiclass classification.

3.5 DISCUSSION

The practical applicability of nonparametric Bayes error estimates to meta-learning and hierarchical classifier design has been investigated. For the BER estimate introduced by Berisha et al. (2016) a bias correction was derived which improves the accuracy of the estimator for classification problems with unequal class priors. Furthermore, a normalization term was proposed which makes the BER estimates comparable in multiclass problems. An expression of the OvR-BER was given which represents the exact Bayes error for a single class in the multiclass problem and it was shown that this error can be efficiently estimated using the HP-estimator as well. A robustness analysis of the HP-estimator was performed which showed the benefit of using orthogonal MSTs in the estimator.

There are many potential applications of the BER estimates to meta-learning problems. Above, several possibilities were explored including the prediction of which pairs of classes are most difficult to distinguish and which individual classes will yield the highest error rate. Preliminary experiments with feature transformations were also performed, which showed that the BER estimates can be a useful tool in determining beneficial transformations before a classifier is trained.

Based on the weighted graph of pairwise BER estimates, a hierarchical multiclass classification method was proposed. The classifier uses a top-down splitting approach to create a tree of binary classification problems which increase in difficulty along the depth of the tree. By using a linear SVM for each classification problem, a hierarchical multiclass SVM was obtained which was named SmartSVM. Extensive simulation studies showed that SmartSVM is often faster than existing approaches and yields competitive predictive performance on several datasets.

Note that the SmartSVM classifier is only one example of how the BER estimates can be used to construct better classification methods. As discussed in Section 3.3, BER estimates could also be used to define class weights in a multiclass classifier. Moreover, the min-cut strategy used for SmartSVM may not be the optimal way to construct the classification tree. Evaluating different approaches to constructing classification hierarchies and other applications of the BER estimates to multiclass classification problems are topics for further research.

Acknowledgements

The computational experiments were performed on the Dutch National LISA Compute Cluster and supported by the Dutch National Science Foundation (NWO). The authors thank SURFsara (www.surfsara.nl) for the support in using the LISA cluster. This research was partially supported by the US Army Research Office, grant W911NF-15-1-0479 and US Dept. of Energy grant DE-NA0002534.

Appendices

3.A One vs. Rest Bayes Error Rate

In this section bounds for the One vs. Rest Bayes error rate will be derived, which measures the error of the Bayes classifier in correctly identifying an individual class.

DEFINITION 3.A.1 (OVR-BER). *Let $f_1, f_2, \ldots, f_K$ and $p_1, p_2, \ldots, p_K$ denote density functions and prior probabilities for the classes 1 through K respectively, with $\sum_c p_c = 1$. Then, the Bayes error rate between a class k and the remaining classes is given by*

$$P_{e,k} = \int\limits_{\max\limits_{l \neq k}\{p_l f_l(\mathbf{x})\} \geq p_k f_k(\mathbf{x})} p_k f_k(\mathbf{x}) \mathrm{d}\mathbf{x} + \sum_{c \neq k} \int\limits_{\max\limits_{l \neq k}\{p_l f_l(\mathbf{x})\} < p_k f_k(\mathbf{x})} p_c f_c(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{3.20}$$

Below it will be shown that the OvR-BER can be bounded using the Friedman-Rafsky statistic in the One-vs-Rest setting, $R(\mathbf{X}_k, \bigcup_{l \neq k} \mathbf{X}_l)$. Let the mixture distribution of the classes $l \neq k$ be given by

$$g_k(\mathbf{x}) = \frac{\sum_{l \neq k} p_l f_l(\mathbf{x})}{\sum_{l \neq k} p_l}, \tag{3.21}$$

with prior probability $p_g = \sum_{l \neq k} p_l$. Then $\mathbf{X}_{g_k} = \bigcup_{l \neq k} \mathbf{X}_l$ can be seen as a draw from this mixture distribution. By Theorem 2 of Berisha et al. (2016) it holds that

$$\tfrac{1}{2} - \tfrac{1}{2}\sqrt{u(f_k, g_k)} \leq \int \min\{p_k f_k(\mathbf{x}), p_g g_k(\mathbf{x})\} \, \mathrm{d}\mathbf{x} \leq \tfrac{1}{2} - \tfrac{1}{2} u(f_k, g_k). \tag{3.22}$$

The following theorem relates this error to the OvR-BER defined above.

THEOREM 3.A.2. *The error rate between class k and the mixture distribution without class k is bounded from above by the OvR-BER,*

$$Q_{e,k} = \int \min\{p_k f_k(\mathbf{x}), p_g g_k(\mathbf{x})\}\,\mathrm{d}\mathbf{x} \leq P_{e,k}. \tag{3.23}$$

Proof. Note that

$$Q_{e,k} = \int\limits_{p_k f_k(\mathbf{x}) \leq p_g g_k(\mathbf{x})} p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad + \int\limits_{p_k f_k(\mathbf{x}) > p_g g_k(\mathbf{x})} p_g g_k(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{3.24}$$

To simplify the notation, introduce the sets

$$T = \left\{\mathbf{x} \in \mathbb{R}^d : p_k f_k(\mathbf{x}) \leq p_g g_k(\mathbf{x})\right\} \tag{3.25}$$

$$S = \left\{\mathbf{x} \in \mathbb{R}^d : p_k f_k(\mathbf{x}) \leq \max_{l \neq k}\{p_l f_l(\mathbf{x})\}\right\} \tag{3.26}$$

and denote their respective complements by T' and S'. Then,

$$Q_{e,k} = \int_T p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int_{T'} p_g g_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{3.27}$$

$$P_{e,k} = \int_S p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int_{S'} p_g g_k(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{3.28}$$

Since $p_g g_k(\mathbf{x}) = \sum_{l \neq k} p_l f_l(\mathbf{x}) \leq \max_{l \neq k} p_l f_l(\mathbf{x})$ it holds that $S \subseteq T$ and $T' \subseteq S'$. Hence,

$$\int_T p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_S p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int_{T \setminus S} p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{3.29}$$

$$\int_{S'} p_g g_k(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_{T'} p_g g_k(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int_{S' \setminus T'} p_g g_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{3.30}$$

However, the sets $T \setminus S$ and $S' \setminus T'$ both equal

$$U = \left\{\mathbf{x} \in \mathbb{R}^d : \max_{l \neq k}\{p_l f_l(\mathbf{x})\} < p_k f_k(\mathbf{x}) \leq p_g g_k(\mathbf{x})\right\}, \tag{3.31}$$

so it follows that

$$Q_{e,k} = P_{e,k} + \int_U p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} - \int_U p_g g_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \leq P_{e,k} \tag{3.32}$$

by definition of the set U. □

This provides a lower bound for $P_{e,k}$ in terms of $u(f_k, g_k)$. What remains to be shown is that $P_{e,k}$ has an upper bound in terms of $u(f_k, g_k)$. No such bound has yet been found. However, the following result can be presented which does bound $P_{e,k}$ from above.

THEOREM 3.A.3. *For a single class k the OvR-BER is smaller than or equal to the sum of the pairwise BER estimates involving class k, that is,*

$$P_{e,k} \leq \sum_{c \neq k} P_e(f_k, f_c). \tag{3.33}$$

Proof. Recall that the OvR-BER for class k is given by

$$P_{e,k} = \int_{p_k f_k(\mathbf{x}) \leq \max\limits_{l \neq k}\{p_l f_l(\mathbf{x})\}} p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad + \sum_{c \neq k} \int_{p_k f_k(\mathbf{x}) > \max\limits_{l \neq k}\{p_l f_l(\mathbf{x})\}} p_c f_c(\mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{3.34}$$

and denote the sum of the pairwise BERs involving k as $F_{e,k}$ given by,

$$\begin{aligned} F_{e,k} &= \sum_{c \neq k} \int \min\{p_k f_k(\mathbf{x}), p_c f_c(\mathbf{x})\}\,\mathrm{d}\mathbf{x} && (3.35)\\ &= \sum_{c \neq k} \int_{p_k f_k(\mathbf{x}) < p_c f_c(\mathbf{x})} p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad + \sum_{c \neq k} \int_{p_k f_k(\mathbf{x}) > p_c f_c(\mathbf{x})} p_c f_c(\mathbf{x})\,\mathrm{d}\mathbf{x}. && (3.36) \end{aligned}$$

Then comparing the first term of $F_{e,k}$ with that of $P_{e,k}$ shows

$$\sum_{c \neq k} \int_{p_k f_k(\mathbf{x}) < p_c f_c(\mathbf{x})} p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad \geq \int_{p_k f_k(\mathbf{x}) \leq \max\limits_{l \neq k}\{p_l f_l(\mathbf{x})\}} p_k f_k(\mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{3.37}$$

since the area of integration on the left is larger than on the right. Similarly,

$$\sum_{c \neq k} \int_{p_k f_k(\mathbf{x}) > p_c f_c(\mathbf{x})} p_c f_c(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad \geq \sum_{c \neq k} \int_{p_k f_k(\mathbf{x}) > \max\limits_{l \neq k}\{p_l f_l(\mathbf{x})\}} p_c f_c(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{3.38}$$

for the same reason. This completes the proof. □

3.B JENSEN-SHANNON BOUND INEQUALITY

In this section a proof is given for the statement that the Henze-Penrose upper bound on the Bayes error rate is tighter than the Jensen-Shannon upper bound derived by Lin (1991). Before presenting the proof, the following lemma is presented.

LEMMA 3.B.1. *For $x, y > 0$ it holds that*

$$x\log\left(1+\frac{y}{x}\right)+y\log\left(1+\frac{x}{y}\right)\geq\frac{4\log(2)xy}{x+y}. \tag{3.39}$$

Proof. Let $t=\frac{y}{x}$ and multiply both sides by $\frac{1}{t}+1>0$, then the inequality becomes

$$\left(\frac{1}{t}+1\right)\log(1+t)+(1+t)\log\left(1+\frac{1}{t}\right)\geq 4\log(2). \tag{3.40}$$

Denote the left hand side by $f(t)$. The proof will now proceed by showing that $f(t)\geq f(1)=4\log(2)$ for all $t>0$. The derivatives of $f(t)$ are given by

$$f'(t)=\log\left(1+\frac{1}{t}\right)-\frac{\log(1+t)}{t^2}, \tag{3.41}$$

$$f''(t)=\frac{2\log(1+t)-t}{t^3}. \tag{3.42}$$

Write the numerator of $f''(t)$ as $g(t)$ such that

$$g(t)=2\log(1+t)-t, \tag{3.43}$$

$$g'(t)=\frac{1-t}{1+t}. \tag{3.44}$$

Then it is clear that $g'(t)>0$ for $0<t<1$ and $g'(t)<0$ for $t>1$. Furthermore $\lim_{t\to 0^+}g(t)=0$ and $\lim_{t\to\infty}g(t)=-\infty$. Thus, it follows that $g(t)$ increases on $0<t<1$ and decreases for $t>1$. Let $t=\alpha>1$ be such that $g(\alpha)=0$, then $g(t)>0$ for $0<t<\alpha$ and $g(t)<0$ for $t>\alpha$.

From this it follows that $f''(t)>0$ for $0<t<\alpha$ and $f''(t)<0$ for $t>\alpha$. Hence, $f'(t)$ is increasing on $0<t<\alpha$ and decreasing for $t>\alpha$. Moreover, $\lim_{t\to 0^+}f'(t)=-\infty$ and $f'(1)=0$. Thus, it follows that $f'(t)$ is negative on $0<t<1$, positive for $t>1$, and attains a maximum at $t=\alpha$ after which it decreases to $\lim_{t\to\infty}f'(t)=0$. Since $\lim_{t\to 0^+}f(t)=\infty$ it follows that $f(t)$ is decreasing on $0<t<1$ and increasing for $t>1$. □

DEFINITION 3.B.2 (KULLBACK-LEIBLER DIVERGENCE). *For probability density functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ the Kullback-Leibler divergence is given by*

$$D_{KL}(f_1\|f_2)=\int f_1(\mathbf{x})\log_2\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}\,\mathrm{d}\mathbf{x}, \tag{3.45}$$

Kullback and Leibler (1951).

DEFINITION 3.B.3 (JENSEN-SHANNON DIVERGENCE). *According to El-Yaniv et al. (1997) the Jensen-Shannon divergence for two probability density functions*

$f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ with prior probabilities p_1 and p_2, can be stated in terms of the Kullback-Leibler divergence as

$$JS(f_1, f_2) = p_1 D_{KL}(f_1 \| M) + p_2 D_{KL}(f_2 \| M) \tag{3.46}$$

with $M(\mathbf{x}) = p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})$ the mixture distribution and $p_2 = 1 - p_1$.

THEOREM 3.B.4. *For $p_1 = p_2 = \frac{1}{2}$ the Henze-Penrose upper bound on the BER is tighter than the Jensen-Shannon upper bound of Lin (1991),*

$$P_e(f_1, f_2) \leq \tfrac{1}{2} - \tfrac{1}{2} u_{HP} \leq \tfrac{1}{2} J, \tag{3.47}$$

where $J = H(p_1) - JS(f_1, f_2)$ with $H(p_1)$ the binary entropy and $JS(f_1, f_2)$ the Jensen-Shannon divergence.

Proof. First, note that with $p_1 = p_2 = \frac{1}{2}$ the binary entropy is given by

$$H(p_1) = -\tfrac{1}{2} \log_2 \left(\tfrac{1}{2}\right) - \tfrac{1}{2} \log_2 \left(\tfrac{1}{2}\right) = 1. \tag{3.48}$$

Second, for the equiprobable case it holds that

$$\tfrac{1}{2} - \tfrac{1}{2} u_{HP}(f_1, f_2) = \int \frac{f_1(\mathbf{x}) f_2(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})} \,\mathrm{d}\mathbf{x}. \tag{3.49}$$

The Jensen-Shannon upper bound can be written as

$$\begin{aligned}
\tfrac{1}{2} J &= \tfrac{1}{2} - \tfrac{1}{4} \int f_1(\mathbf{x}) \log_2 \frac{2 f_1(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})} \,\mathrm{d}\mathbf{x} && (3.50)\\
&\quad - \tfrac{1}{4} \int f_2(\mathbf{x}) \log_2 \frac{2 f_2(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})} \,\mathrm{d}\mathbf{x} \\
&= \tfrac{1}{4} \int f_1(\mathbf{x}) + f_2(\mathbf{x}) \,\mathrm{d}\mathbf{x} - \tfrac{1}{4} \int f_1(\mathbf{x}) \log_2 \frac{2 f_1(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})} \,\mathrm{d}\mathbf{x} && (3.51)\\
&\quad - \tfrac{1}{4} \int f_2(\mathbf{x}) \log_2 \frac{2 f_2(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})} \,\mathrm{d}\mathbf{x} \\
&= \tfrac{1}{4} \int f_1(\mathbf{x}) \left[1 - \log_2 \frac{2 f_1(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})} \right] \mathrm{d}\mathbf{x} && (3.52)\\
&\quad + \tfrac{1}{4} \int f_2(\mathbf{x}) \left[1 - \log_2 \frac{2 f_2(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})} \right] \mathrm{d}\mathbf{x}. \\
&= \tfrac{1}{4} \int f_1(\mathbf{x}) \log_2 \left(1 + \frac{f_2(\mathbf{x})}{f_1(\mathbf{x}} \right) + f_2(\mathbf{x}) \log_2 \left(1 + \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \right) \mathrm{d}\mathbf{x} && (3.53)
\end{aligned}$$

By Lemma 3.B.1 it follows that

$$f_1(\mathbf{x}) \log_2 \left(1 + \frac{f_2(\mathbf{x})}{f_1(\mathbf{x}} \right) + f_2(\mathbf{x}) \log_2 \left(1 + \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \right) \geq \frac{4 f_1(\mathbf{x}) f_2(\mathbf{x})}{f_1(\mathbf{x}) + f_2(\mathbf{x})}, \tag{3.54}$$

and therefore

$$\tfrac{1}{2}J \geq \tfrac{1}{4}\int \frac{4f_1(\mathbf{x})f_2(\mathbf{x})}{f_1(\mathbf{x})+f_2(\mathbf{x})}\,\mathrm{d}\mathbf{x} = \tfrac{1}{2} - \tfrac{1}{2}u_{HP}(f_1,f_2). \tag{3.55}$$

$\square$

3.C Additional Simulation Results

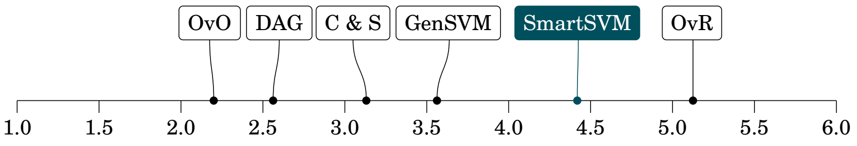
In this section some additional simulation results are presented for the SmartSVM experiments presented in Section 3.4. Table 3.4 shows the average time per hyperparameter configuration for each of the methods. This is especially useful for comparing GenSVM (Van den Burg and Groenen, 2016) with the other methods, as it has a larger set of hyperparameters to consider.

A commonly used tool to summarize results of simulation experiments is to use rank plots (Demšar, 2006). For each dataset the methods are ranked, with the best method receiving rank 1 and the worst method receiving rank 6 (since there are 6 methods in this experiment). In case of ties fractional ranks are used. By averaging the ranks over all datasets, a visual summary of the results can be obtained. Figures 3.10(a), 3.10(b) and 3.10(c) show these average ranks for predictive performance, total training time, and average training time respectively.

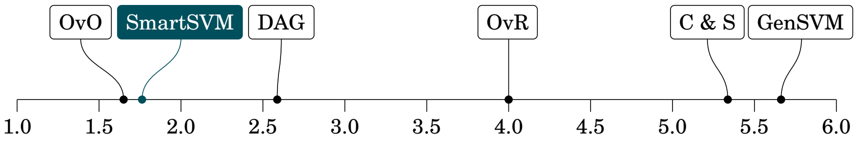
The ordering of OvO and SmartSVM in the rank plots for training time may seem counterintuitive, considering that SmartSVM is more often the fastest method. This can be explained by the fact that in the cases where SmartSVM is slower than OvO it is usually also slower than DAG. In contrast, where SmartSVM is the fastest method OvO is usually the second fastest method. Because of this, SmartSVM obtains a slightly higher average rank than OvO.

TABLE 3.4 – *Average training time per hyperparameter configuration in seconds, averaged over the five nested CV folds. Minimal values per dataset are underlined.*

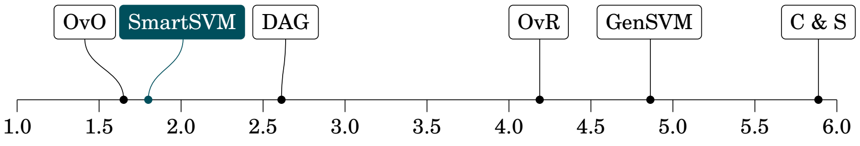
Dataset	C & S	DAG	GenSVM	OvO	OvR	SmartSVM
abalone	702	7.712	254	7.067	11.282	3.960
fars	8329	164	601	151	296	184
flare	24.606	0.369	1.467	0.344	0.431	0.263
krkopt	4526	38.337	165	35.813	73.053	35.627
letter	1668	21.438	131	20.076	94.977	34.454
nursery	119	3.932	3.985	3.617	4.994	3.133
optdigits	4.630	1.053	92.441	0.971	1.278	0.590
pageblocks	27.550	1.408	5.852	1.326	2.559	1.504
pendigits	132	1.589	19.273	1.498	7.965	2.872
satimage	273	3.931	7.494	3.685	9.918	3.942
segment	19.846	0.376	4.509	0.370	0.985	0.285
shuttle	403	30.042	42.744	29.571	105	43.928
texture	30.954	1.816	43.200	1.727	3.665	1.028
winered	37.179	0.646	1.587	0.586	0.966	0.465
winewhite	135	3.115	7.514	2.903	3.732	1.954
yeast	63.655	0.877	3.327	0.807	1.272	0.699



(a) Predictive performance



(b) Total training time



(c) Average training time

FIGURE 3.10 – *Rank plots of classifier performance in the simulation study. Figure (a) shows the average ranks for out-of-sample predictive performance as measured by the ARI. Figures (b) and (c) respectively show the average ranks for total training time and average training time per hyperparameter configuration.*

II

Regularized Regression

4

SparseStep: Approximating the Counting Norm for Sparse Regularization

G.J.J. van den Burg, P.J.F. Groenen, and A. Alfons

Abstract

The SparseStep algorithm is presented for the estimation of a sparse parameter vector in the linear regression problem. The algorithm works by adding an approximation of the exact ℓ_0 norm as a constraint on the model parameters, and iteratively strengthening this approximation to arrive at a sparse solution. Theoretical analysis of the penalty function shows that the estimator yields unbiased estimates of the parameter vector. An iterative majorization algorithm is derived which has a straightforward implementation reminiscent of ridge regression. In addition, the SparseStep algorithm is compared with similar methods through a rigorous simulation study which shows it often outperforms existing methods in both model fit and prediction accuracy.

This chapter is based on Van den Burg et al. (2017).

4.1 Introduction

In many modeling problems it is desirable to restrict the number of nonzero elements in the parameter vector to reduce the model complexity. This can be done by using ℓ_0 norm regularization to allow only a limited number of coefficients in the model. However, achieving this so-called sparsity in the parameters for the regression problem is known to be NP-hard (Natarajan, 1995). Many alternatives have been presented in the literature which approximate the true sparse solution by applying shrinkage to the parameter vector, such as the lasso estimator (Tibshirani, 1996). However, this shrinkage can underestimate the true effect of explanatory variables on the model outcome. Here, an algorithm is presented which creates sparse model estimates but does not apply shrinkage to the parameter estimates. This is achieved by iteratively refining an approximation to the exact ℓ_0 norm which counts the number of nonzero elements in a vector.

Traditional methods for solving the exact sparse linear regression problem include best subset selection, forward and backward stepwise regression, and forward stagewise regression. These approaches may not always be feasible for problems with a large number of predictors, and may display a high degree of variance with out-of-sample predictions (Hastie et al., 2009). Alternatively, penalized least-squares methods add a regularization term to the regression problem, to curb variability through shrinkage or induce sparsity, or both. Of the many more recent approaches to this problem, the most well-known are perhaps the SCAD penalty (Fan and Li, 2001) and the MC+ penalty (Zhang, 2010). In both of these approaches, a penalty is added such that the overall size of the model parameters can be controlled.

Figure 4.1 shows an illustration of the different penalty functions discussed above. Note that all penalty functions are symmetric around zero, including the SparseStep penalty introduced below. It can be seen that the shapes of the SCAD and MC+ penalties closely resemble each other. The different shapes of the penalty function for the SCAD and MC+ penalties are due to the hyperparameter a, which can be optimized over for a given dataset.

In this work, the SparseStep regression algorithm will be presented. This algorithm directly targets the regression problem penalized with the ℓ_0 norm. This is done by replacing the exact norm with a smooth approximation and iteratively refining the approximation until it matches the ℓ_0 norm exactly (see Figure 4.1(d) for an illustration). This strategy to approximate the exact ℓ_0 norm has been used previously by Mohimani et al. (2009, 2010) for the SL0 algorithm in the compressed sensing and signal processing literature. Here, the smoothing approach is applied to the regression context and tested against the

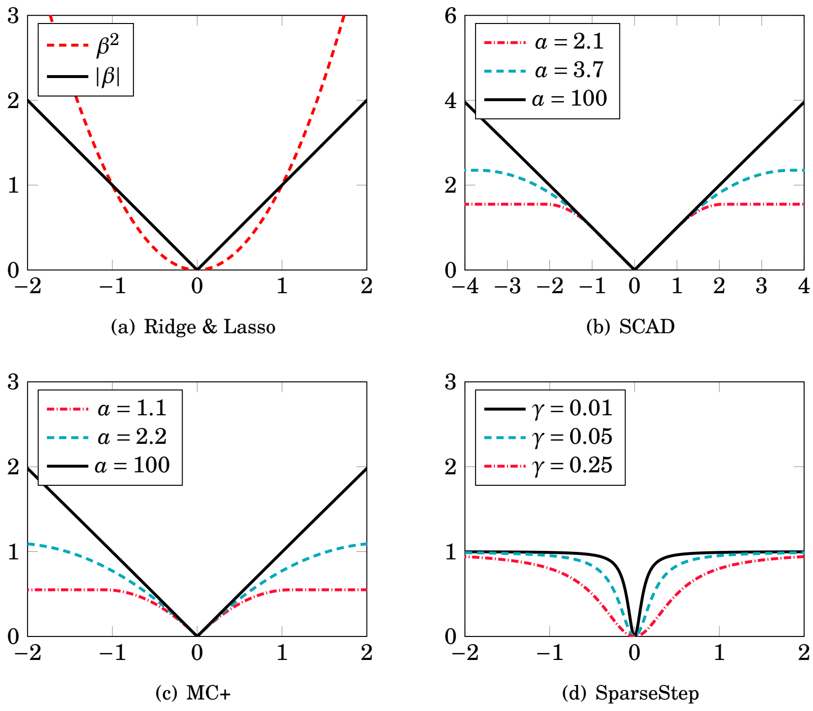


FIGURE 4.1 – *Illustrations of different penalty functions, with regularization parameter $\lambda = 1$ where applicable. The SCAD and MC+ penalties have a tuning parameter a that modifies the behavior of the penalty. With the exception of the Ridge penalty all penalties can induce sparsity in the parameter vector. All penalties are symmetric around the origin. Note that out of all penalties shown here only the Ridge and SparseStep penalties have continuous first derivatives at the origin. For the SCAD penalty the axes are rescaled to accommodate the requirement that $a > 2$ for this penalty.*

commonly used alternatives mentioned above. Moreover, instead of the Gaussian smoothing function used by Mohimani et al. (2009) a fractional smoothing penalty is used which they analyze theoretically but do not verify empirically. Finally, in the SparseStep algorithm an Iterative Majorization strategy is derived which avoids the projection step of the SL0 algorithm, thereby potentially improving performance.

This chapter is organized as follows. In Section 4.2 the theory behind the SparseStep penalty is introduced and analyzed. Section 4.3 derives the SparseStep algorithm using the Iterative Majorization technique, and describes the implementation of the algorithm. Experiments comparing SparseStep with existing methods are described extensively in Section 4.4. Section 4.5 concludes.

4.2 Theory

Below the theory of norms and regularized regression is briefly reviewed, after which the SparseStep norm approximation is presented and some theoretical properties of this approximation are analyzed. Finally, the loss function is presented for regularized regression using the SparseStep norm approximation.

4.2.1 *Norms*

The ℓ_p norm of a vector $\boldsymbol{\beta} \in \mathbb{R}^m$ is defined as

$$\|\boldsymbol{\beta}\|_p \equiv \left(\sum_{j=1}^{m} |\beta_j|^p \right)^{1/p}. \tag{4.1}$$

For $p = 2$, the well known Euclidean norm is obtained, whereas for $p = 1$ the distance measured is known as the Manhattan distance. In regularized regression it is common to add the $\|\boldsymbol{\beta}\|_p^p$ norm as a penalty term (i.e. the ℓ_p norm raised to the power p). When $p = 0$ this function seizes to be a proper norm due to the lack of homogeneity and it equals the number of nonzero elements of $\boldsymbol{\beta}$ (Peetre and Sparr, 1972, Donoho, 2006). However, it is common to abuse terminology and refer to this function as the ℓ_0 norm or the counting norm, and in this work this custom will also be used. The ℓ_0 norm of a vector $\boldsymbol{\beta} \in \mathbb{R}^m$ is thus defined as

$$\|\boldsymbol{\beta}\|_0^0 = \sum_{j=1}^{m} \pi[\beta_j \neq 0], \tag{4.2}$$

where $\pi(\cdot)$ is an indicator function which is 1 if it's argument is true, and 0 otherwise. The ℓ_0 norm is shown graphically in Figure 4.2 for the two dimensional case. It can be seen that this norm is discontinuous and nonconvex.

4.2.2 *Regularization*

Let $\mathscr{D} = \{(\mathbf{x}_i', y_i)\}_{i=1,\ldots,n}$ denote the data for the regression problem, with explanatory variables $\mathbf{x}_i = (x_{i1}, \ldots, x_{im})' \in \mathbb{R}^m$ and outcome $y_i \in \mathbb{R}$. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ denote the data matrix with rows $\mathbf{x}_i'$. Assume that the vector of outcomes $\mathbf{y} \in \mathbb{R}^n$ is centered, so that the intercept term can be ignored. The least-squares regression problem can then be written as

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2. \tag{4.3}$$

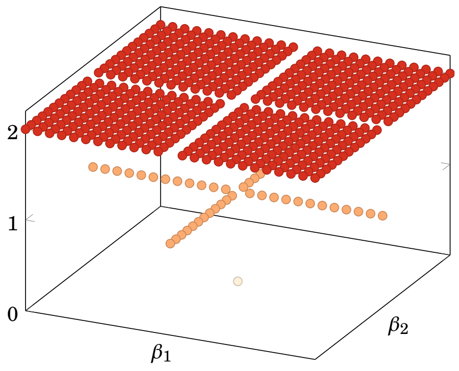


FIGURE 4.2 – *Illustration of the exact ℓ_0 norm in two dimensions. Note the strong discontinuities along both axes, as indicated by the color of the points. The variables β_1 and β_2 may vary continuously along the axes, but for illustration purposes the ℓ_0 norm is computed here on a grid of values.*

The ℓ_p-*regularized* least-squares problem can be defined through the loss function

$$L(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{m} |\beta_j|^p, \tag{4.4}$$

with $\lambda \geq 0$ a regularization parameter. Two well-known special cases of these regularized least-squares problems are ridge regression (Hoerl and Kennard, 1970) corresponding to $p = 2$, and the lasso estimator (Tibshirani, 1996) corresponding to $p = 1$. When $p = 0$ is used the regularization term turns into the ℓ_0 norm of the β_j. Note that in this case no shrinkage of the β_j occurs because only the number of nonzero β_j is controlled and not their size.

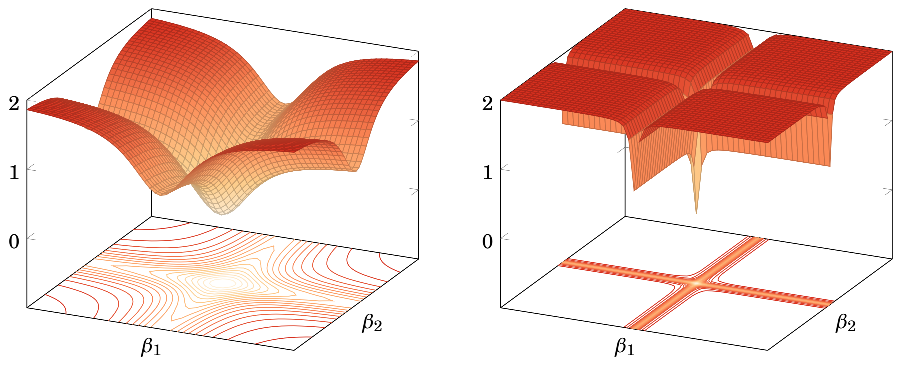
4.2.3 *Norm Approximation*

Recently, an approximation to the ℓ_0 norm was proposed by De Rooi and Eilers (2011), where the indicator function of an element β_j is approximated as

$$\pi[\beta_j \neq 0] \approx \frac{\beta_j^2}{\beta_j^2 + \gamma^2}, \tag{4.5}$$

where $\gamma \geq 0$ is a positive constant[1]. Note that if $\gamma = 0$ the approximation becomes exact. By decreasing the value of the γ parameter the approximation of the ℓ_0

[1]For consistency with the remainder of the chapter the current definition of γ deviates from that of De Rooi and Eilers (2011). In contrast to their definition of the approximation, the square of γ is used here.



(a) $\gamma^2 = 0.3$ (b) $\gamma^2 = 0.05$

FIGURE 4.3 – *Three-dimensional illustrations of the norm approximation proposed by De Rooi and Eilers (2011) for two different values of γ. It can be seen that for a smaller value of γ the approximation more closely resembles the exact ℓ_0 norm illustrated in Figure 4.2.*

norm becomes increasingly more accurate. Figures 4.3(a) and 4.3(b) show the approximation for both large and small values of γ, respectively. It can be seen that for decreasing values of γ the approximation indeed converges to the exact ℓ_0 norm shown in Figure 4.2.

In the following, the approximation described above will be used to define the *SparseStep penalty* as

$$P_\lambda\left(\beta_j\right) = \lambda \frac{\beta_j^2}{\beta_j^2 + \gamma^2}. \tag{4.6}$$

This penalty function is symmetric around 0 and is continuously differentiable.

In a similar analysis for the SCAD penalty, Fan and Li (2001) describe a sufficient condition for a penalty function to achieve unbiasedness of the parameter estimates. Fan and Li (2001) argue that for unbiasedness of the estimates of large true values of β_j it is sufficient that the penalty term is zero for large values of $|\beta_j|$. The derivative of the SparseStep penalty is

$$P'_\lambda\left(\beta_j\right) = \lambda \frac{2\gamma^2 \beta_j}{\left(\beta_j^2 + \gamma^2\right)^2}, \tag{4.7}$$

and since

$$\lim_{|\beta_j|\to\infty} P'_\lambda\left(|\beta_j|\right) = \lim_{|\beta_j|\to\infty} \lambda \frac{2\gamma^2 |\beta_j|}{\left(|\beta_j|^2 + \gamma^2\right)^2} = 0, \tag{4.8}$$

we conclude that the SparseStep penalty results in unbiased estimates.

Additionally, Fan and Li (2001) derive sufficient conditions for a penalty function to have the Oracle Property. This property means that under certain regularity conditions a method correctly identifies the sparsity in the predictor variables correctly, as the number of observations goes to infinity. One sufficient condition for this is that the derivative of the penalty function should be positive at the origin. This does not hold exactly for SparseStep, but does hold in an arbitrarily small region around the origin, due to the value of γ. Further research is necessary to establish whether or not the Oracle Property holds for SparseStep, but it is hypothesized that this is indeed the case.

The above leads naturally to the formulation of the SparseStep regression problem, with loss function

$$L(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{m} \frac{\beta_j^2}{\beta_j^2 + \gamma^2}. \tag{4.9}$$

In the next section, an Iterative Majorization algorithm will be derived for minimizing this loss function.

4.3 Methodology

With the theoretical underpinnings of SparseStep established above, it is now possible to derive the optimization algorithm necessary to minimize the SparseStep regression loss function. The approach used here is that of the *iterative majorization* algorithm. A brief introduction is given first, followed by the derivation of the SparseStep algorithm.

4.3.1 *Iterative Majorization*

The Iterative Majorization (IM) algorithm is a general optimization algorithm based on surrogate functions, first described by Ortega and Rheinboldt (1970). It is also known as the Majorization Minimization algorithm and is a generalization of the popular Expectation Maximization algorithm (see e.g. Hunter and Lange, 2004). A brief description of the algorithm follows.

Let $f : \mathcal{X} \rightarrow \mathcal{Z}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Z} \subseteq \mathbb{R}$ be the function that needs to be optimized. Construct a majorizing function $g : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Z}$ such that

$$f(y) = g(y, y), \tag{4.10}$$

$$f(x) \leq g(x, y) \text{ for all } x, y \in \mathcal{X}, \tag{4.11}$$

where y is the so-called *supporting point*. Differentiability of f at y implies that $\nabla f(y) = \nabla g(y,y)$. Given a majorizing function g the following procedure results in a stationary point of f:

1. Let $y = x_0$, with x_0 a starting point
2. Minimize $g(x,y)$ and let $x^+ = \arg\min g(x,y)$
3. Stop if a stopping criterion is reached, otherwise let $y = x^+$ and go to step 2.

This procedure yields a guaranteed descent algorithm where $f(x^+) \leq f(y)$, with a linear convergence rate (De Leeuw, 1994). However, a well-known property of the IM algorithm is that in the first few iterations often large improvements in the loss function can be made (Havel, 1991). This property makes it ideally suited for the SparseStep algorithm described below. Generally, a sufficiently simple functional form is chosen for the majorizing function such that Step 2 in the above procedure can be done swiftly.

4.3.2 *Majorization Derivation*

The majorizing function of the SparseStep loss function will be derived here. For ease of notation, let $f(x)$ denote the penalty function, with $x \in \mathbb{R}$, and $g(x,y)$ the majorizing function, with $x, y \in \mathbb{R}$. Then,

$$f(x) = \frac{x^2}{x^2 + \gamma^2}. \tag{4.12}$$

Since $f(x)$ is a differentiable and even function (i.e. $f(x) = f(-x)$), the sharp quadratic majorization function derived in De Leeuw and Lange (2009) can be used, which gives

$$g(x,y) = \frac{f'(y)}{2y}(x^2 - y^2) + f(y) = \frac{\gamma^2 x^2 + y^4}{\left(y^2 + \gamma^2\right)^2}. \tag{4.13}$$

The only requirement for this majorization function to be valid is that $h(x) = f'(x)/x$ is decreasing on the interval $(0,\infty)$. Since

$$h(x) = \frac{f'(x)}{x} = \frac{2\gamma^2}{\left(x^2 + \gamma^2\right)^2} \tag{4.14}$$

it is straightforward to show that for any $u, v \in (0,\infty)$ it holds that if $u \leq v$ then $h(u) \geq h(v)$. Thus $h(x)$ is a monotonically decreasing function on the interval $(0,\infty)$ and the majorization function is valid. Figure 4.4 shows the majorizing function and the SparseStep penalty function for different values of the supporting point.

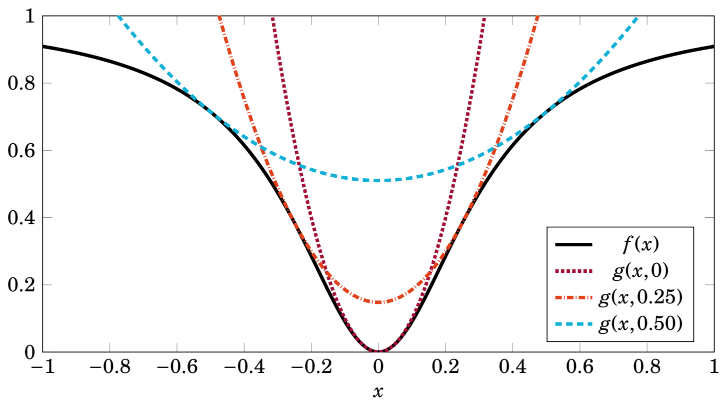


FIGURE 4.4 – *Illustration of the SparseStep penalty function $f(x)$ and the majorizing function $g(x,y)$ for various values of the supporting point y, with $\gamma^2 = 0.1$.*

Recall that the SparseStep loss function is given by

$$L(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{m} \frac{\beta_j^2}{\beta_j^2 + \gamma^2}. \tag{4.15}$$

Let α_j denote the previous value of β_j in the IM algorithm (the supporting point). Then, using the majorizing function derived above it is clear that the following inequality holds

$$L(\boldsymbol{\beta}) \leq \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{m} \frac{\gamma^2 \beta_j^2 + \alpha_j^4}{(\alpha_j^2 + \gamma^2)^2} = G(\boldsymbol{\beta}, \boldsymbol{\alpha}), \tag{4.16}$$

where $G(\boldsymbol{\beta}, \boldsymbol{\alpha})$ denotes the majorizing function of $L(\boldsymbol{\beta})$. Taking the derivative of $G(\boldsymbol{\beta}, \boldsymbol{\alpha})$ with respect to $\boldsymbol{\beta}$ yields an explicit expression for the update of $\boldsymbol{\beta}$ in the IM algorithm. Before taking the derivative of the majorizing function however, let us define

$$\boldsymbol{\Omega}_{jj} = \frac{\gamma^2}{(\alpha_j^2 + \gamma^2)^2} \tag{4.17}$$

$$\delta_j = \alpha_j^2 / \gamma, \tag{4.18}$$

such that $\boldsymbol{\Omega}$ is an $m \times m$ diagonal matrix with elements $\boldsymbol{\Omega}_{jj}$, and $\boldsymbol{\delta} \in \mathbb{R}^m$ a vector with elements δ_j.

With these definitions, the regularization term becomes

$$\lambda \sum_{j=1}^{m} \frac{\gamma^2 \beta_j^2 + \alpha_j^4}{(\alpha_j^2 + \gamma^2)^2} = \lambda(\boldsymbol{\beta}'\boldsymbol{\Omega}\boldsymbol{\beta} + \boldsymbol{\delta}'\boldsymbol{\Omega}\boldsymbol{\delta}). \tag{4.19}$$

By expanding the norm and using this form for the regularization term, it is possible to write $G(\boldsymbol{\beta}, \boldsymbol{\alpha})$ as

$$G(\boldsymbol{\beta}, \boldsymbol{\alpha}) = \mathbf{y}'\mathbf{y} - 2\mathbf{y}'\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}'(\mathbf{X}'\mathbf{X} + \lambda\boldsymbol{\Omega})\boldsymbol{\beta} + \lambda\boldsymbol{\delta}'\boldsymbol{\Omega}\boldsymbol{\delta}. \tag{4.20}$$

Taking the derivative to $\boldsymbol{\beta}$ and setting this to zero, yields

$$-2\mathbf{X}'\mathbf{y} + 2(\mathbf{X}'\mathbf{X} + \lambda\boldsymbol{\Omega})\boldsymbol{\beta} = 0 \tag{4.21}$$

Thus, the update of the majorization algorithm is simply

$$\boldsymbol{\beta} = \left(\mathbf{X}'\mathbf{X} + \lambda\boldsymbol{\Omega}\right)^{-1}\mathbf{X}'\mathbf{y}. \tag{4.22}$$

Since $\boldsymbol{\Omega}$ is a diagonal matrix this expression is remarkably similar to the solution of the ridge regression problem, in which $\boldsymbol{\Omega}$ is simply the identity matrix.

4.3.3 *SparseStep Algorithm*

With the derivation of the IM algorithm for minimizing the SparseStep regression loss function, it is now possible to formulate the SparseStep algorithm. To avoid local minima, the SparseStep penalty is introduced slowly by starting with a large value of γ, so that the penalty is very smooth and behaves like a ridge penalty. Then, the γ value is reduced so that the irregularity and nonconvexity is introduced slowly. The value of γ is reduced until it is close to zero. By slowly introducing the nonsmoothness in the penalty and taking only a few steps of the IM algorithm for each γ, the SparseStep algorithm aims to avoid local minima and tries to reach the global minimum of the regression problem with the ℓ_0 norm penalty. Pseudocode for SparseStep regression is given in Algorithm 4.1.

The algorithm starts by initializing $\boldsymbol{\beta}$ and γ from given values $\boldsymbol{\beta}_0$ and γ_0 respectively. For each value of γ the parameter estimates are updated t_{max} times using the IM algorithm. Subsequently γ is reduced by a factor γ_{step}. This process is continued until γ reaches a provided stopping value γ_{stop}. In the end, sufficiently small elements of $\boldsymbol{\beta}$ are set to absolute zero by comparing to a small constant ϵ. This is done to avoid numerical precision errors, and can be a method for enhancing the sparsity inducing properties of SparseStep. The value of ϵ and

Algorithm 4.1. SparseStep Regression

```
1: function SparseStep($\mathbf{X}, \mathbf{y}, \boldsymbol{\beta}_0, \gamma_0, \gamma_{stop}, \gamma_{step}, t_{max}, \epsilon$)
2:     $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_0$
3:     $\gamma \leftarrow \gamma_0$
4:     while $\gamma > \gamma_{stop}$ do
5:         for $t = 1$ to $t_{max}$ do
6:             Construct $\boldsymbol{\Omega}$ according to (4.17)
7:             $\boldsymbol{\beta} \leftarrow \left(\mathbf{X}'\mathbf{X} + \lambda\boldsymbol{\Omega}\right)^{-1}\mathbf{X}'\mathbf{y}$
8:         end for
9:         $\gamma \leftarrow \gamma/\gamma_{step}$
10:    end while
11:    for $j \in \{1, \ldots, m\}$ do
12:        if $|\beta_j| < \epsilon$ then
13:            $\beta_j \leftarrow 0$
14:        end if
15:    end for
16:    return $\boldsymbol{\beta}$
17: end function
```

that of γ_{stop} are related. Note that in an actual implementation the matrices $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}'\mathbf{y}$ should be cached for computational efficiency.

4.4 Experiments

To verify the performance of the SparseStep algorithm in correctly identifying the nonzero predictor variables in a regression problem, a simulation study was performed. The aim of this simulation study is to mimic as much as possible a practical setting where a researcher is interested in both the predictive accuracy of a regression model and the correct identification of variables with nonzero coefficients. Moreover, this simulation study allows verification of the performance of the SparseStep algorithm for datasets with varying statistical properties such as the number of variables, the signal-to-noise ratio (SNR), the correlation between the variables, and the degree of sparsity in the true coefficient vector.

A second goal of this simulation study is to compare the performance of the SparseStep algorithm with existing regression methods. These competing methods are: ordinary least-squares, lasso (Tibshirani, 1996), ridge regression (Hoerl and Kennard, 1970), SCAD (Fan and Li, 2001), and MC+ (Zhang, 2010). Thus, the focus here is on comparing SparseStep with other penalized regression methods, including some that induce sparsity through penalization. In order to accurately evaluate the predictive accuracy of these methods and to find the best regularization parameter for each method, separate training and testing datasets will be used. The procedure is then to find for each method the regularization parameter which performs best on the training dataset during 10-fold cross-validation as measured by the mean squared error (MSE) of the outcome variable. Next, each method is trained one more time on the entire training dataset using this optimal regularization parameter and the obtained model is used to predict the test dataset. Predictive accuracy is measured using the MSE on the test dataset.

The accuracy of the estimated parameter vector $\hat{\boldsymbol{\beta}}$ will be evaluated on two measures: the mean squared error with respect to the true $\boldsymbol{\beta}$, and the *sparsity hit rate*. The sparsity hit rate is calculated simply as the sum of the correctly identified zero elements of $\boldsymbol{\beta}$ and the correctly identified nonzero elements, divided by the total number of elements of $\boldsymbol{\beta}$.

For the simulation study the following data generating process will be used. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ denote the data matrix drawn from a multivariate normal distribution with mean vector $\boldsymbol{\mu} \in \mathbb{R}^m$ and correlation matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times m}$, such that the rows $\mathbf{x}_i' \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The data matrix $\mathbf{X}$ was scaled such that each column had mean 0 and unit variance. In all simulated datasets $\boldsymbol{\mu}$ is drawn from an m-dimensional standard uniform distribution. For the correlation matrix $\boldsymbol{\Sigma}$ three different scenarios are used: uncorrelated, constantly correlated, and noise correlated. In the uncorrelated case the $\boldsymbol{\Sigma}$ matrix is simply the identity matrix, in the constantly correlated case all variables have a correlation of .5 with each other, and in the noise correlated case a correlation matrix is generated by adding realistic noise to the identity matrix using the method of Hardin et al. (2013)[2]. Next, a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^m$ is drawn from a uniform distribution with elements $\beta_j \in [-1, 1]$. The last z elements of $\boldsymbol{\beta}$ are set to zero to simulate sparsity. Finally, to obtain realistic data with a known signal-to-noise ratio, the simulated outcome variable $\mathbf{y}$ is calculated as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}, \tag{4.23}$$

where $\mathbf{e} \in \mathbb{R}^n$ is a noise term which contains n elements drawn from a univariate normal distribution with mean zero and standard deviation such that the SNR given by $\boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta}/\mathbf{e}'\mathbf{e}$ is as desired.

Table 4.1 gives an overview of how the different parameters of the data generating process were varied among datasets. Using these parameters a total of 180 datasets were generated. For all datasets the number of instances n was set to 30,000, which was then split into 20,000 instances in the training dataset and 10,000 in the testing dataset. Note that the degree of sparsity in the table is expressed as a percentage of the number of variables m. In practice, the number of zeroes in $\boldsymbol{\beta}$ corresponds to $z = \lfloor m \cdot \zeta / 100 \rfloor$, where ζ is a number taken from the second row of the table.

[2]This corresponds to Algorithm 4 in the paper of Hardin et al. (2013), using the default parameters of $\varepsilon = 0.01$ and $M = 2$ (in their notation).

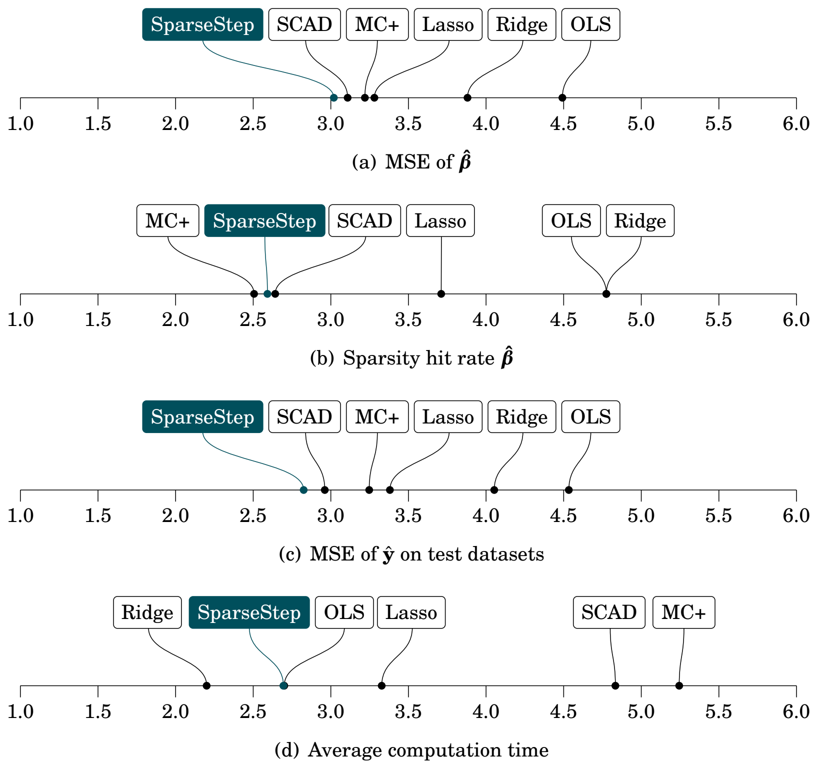
TABLE 4.1 – *Overview of the values for different parameters in the data generating process of the simulation study. Datasets were generated for each possible combination of these values, resulting in 180 datasets.*

Parameter	Values
Variables (m)	$\{10, 50, 100, 500\}$
Sparsity ($\zeta\%$)	$\{0, 25, 50, 75, 95\}$
SNR	$\{0.1, 1.0, 10.0\}$
Correlation	{uncorrelated, constant, noise}

The simulation study was set up such that each method was trained on exactly the same cross-validation sample when the same value of λ was supplied. The grid of λ parameters for the regularized methods came from a logarithmically spaced vector of 101 values between 2^{-15} and 2^{15}. For SparseStep, the input parameters were chosen as $\gamma_0 = 10^6$, $\gamma_{stop} = 10^{-8}$, $\gamma_{step} = 2$, $t_{max} = 2$, $\epsilon = 10^{-7}$, and $\boldsymbol{\beta}_0 = \mathbf{0}$ (see Algorithm 4.1). Default input parameters where chosen for the other methods where applicable. The R language (R Core Team, 2015) was used for the SCAD and MC+ methods, with SCAD implemented through the ncvreg package (Breheny and Huang, 2011), and MC+ through the SparseNet package (Mazumder et al., 2011). The other methods were implemented in the Python language (Van Rossum, 1995) using the scikit-learn package (Pedregosa et al., 2011). For the MC+ penalty the secondary regularization parameter a was optimized for the training dataset using the CV implementation of the SparseNet package. For SCAD a was set to 3.7 as this is the default of Fan and Li (2001).

To determine statistically significant differences between the performance of each of the methods, recommendations on benchmarking machine learning methods will be used as formulated by Demšar (2006). Specifically, *rank tests* will be applied to evaluate whether SparseStep outperforms other methods significantly. For each dataset, fractional ranks are calculated for each performance measure with a smaller rank indicating a better performance. Methods are considered to have equal performance if the difference on a performance metric is smaller than 10^{-4}. A Friedman rank test can be done on the calculated ranks to test for equal performance of the methods (Friedman, 1937, 1940), and Holm's step down procedure can be used to test for significant differences between SparseStep and other methods (Holm, 1979).

Figure 4.5 shows the average ranks of the six evaluated methods on four different metrics. From Figure 4.5(a), which shows the average ranks on the MSE of $\hat{\boldsymbol{\beta}}$, it can be seen that SparseStep is most often the best method for fitting $\boldsymbol{\beta}$, followed closely by SCAD, MC+, and the Lasso. The sparsity hit rate of $\hat{\boldsymbol{\beta}}$ is on average the best for the MC+ penalty, followed by SparseStep and SCAD, as illustrated in Figure 4.5(b). For the out-of-sample performance on the test data,



(a) MSE of $\hat{\boldsymbol{\beta}}$

(b) Sparsity hit rate $\hat{\boldsymbol{\beta}}$

(c) MSE of $\hat{\mathbf{y}}$ on test datasets

(d) Average computation time

FIGURE 4.5 – *Rank plots showing the average performance of the six evaluated methods on the 180 simulated datasets. Points on the line show the fractional rank of the methods averaged over all datasets, with a smaller rank indicating a better performance. It can therefore be seen that SparseStep performs very favorably in both predicting the size and sparsity of* $\boldsymbol{\beta}$*, as well as on predicting the outcome out-of-sample. Graph* (d) *shows the rank graph for the average computation time for each method.*

shown in 4.5(c), a similar order of the methods can be observed as for the MSE on $\hat{\boldsymbol{\beta}}$, although the difference between SCAD and MC+ is larger here. SparseStep again outperforms the other methods on this measure.

Computation time was also measured for each method on each dataset. The rank plot of the average computation time per dataset is given in Figure 4.5(d). It can be seen that SparseStep performs well on average. The average computation time of SparseStep for a single value of λ is comparable to computing a single OLS solution. An important caveat with regards to the computation times is that in order to have the same CV splits for each method with a certain λ, the path algorithms of the Lasso, SCAD, and MC+ penalty could not be used. The computation time of these methods is therefore overestimated.

Apart from the ranks averaged over all datasets, it is also interesting to look at how often a method is the best method on a dataset and how often it is the worst method. Looking at the MSE of $\hat{\boldsymbol{\beta}}$, Ridge is most often the best method, followed closely by the penalized methods. As expected OLS is most often the worst method in this regard. Next, SparseStep most often obtains the highest sparsity hit rate of $\boldsymbol{\beta}$, on 30 out of 180 datasets. MC+ is the best method on this metric on 26 datasets, and SCAD on 11. Finally, when considering the MSE on the outcome of the test dataset all the penalized methods are the best method with similar frequency. An exception to this is OLS, which is the best method on only 7 datasets and the worst method on 52 of them. MC+ is the worst method on 32 datasets, whereas SparseStep is the worst method the smallest number of times, on only 6 datasets. Clearly, OLS and Ridge only perform well on datasets without sparsity in $\boldsymbol{\beta}$. For the other methods no clear relationship between the dataset characteristics and the performance of the method could be found.

As suggested by Demšar (2006) an F-test can be done on the average ranks to evaluate if significant differences exist between the different methods. This is the case for the four measures discussed above, all with p-value < 0.0001. Furthermore, Holm's procedure can be performed to uncover significant differences between the methods and a reference method, in this case SparseStep. From this it is found that on the performance metrics other than computation time, SparseStep significantly outperforms OLS and Ridge, but that the difference between SparseStep and the other penalized methods is not significant at the 5% level. On the computation time metric SparseStep significantly outperforms SCAD and MC+ at the 5% level, but the caveat mentioned above should be taken into account here. The lack of a significant difference between SparseStep and SCAD and MC+ on the other metrics can be due to either a lack of any theoretical difference, or an insufficient number of datasets in the simulation study.

4.5 Discussion

This chapter presents the SparseStep algorithm which induces sparsity in the regression problem by iteratively improving an approximation of the ℓ_0 norm. An iterative majorization algorithm has been derived which is straightforward to implement. The practical relevance of SparseStep is evaluated through a thorough simulation study on 180 datasets with varying characteristics. The results indicate that SparseStep often outperforms existing methods, both in identification of the parameter vector and in out-of-sample prediction. Future research will focus on the theoretical properties of the SparseStep algorithm, such as the criteria for convergence to a global optimum.

Acknowledgements

The computational tests of this research were performed on the Dutch National LISA cluster, and supported by the Dutch National Science Foundation (NWO). The authors thank SURFsara (www.surfsara.nl) for the support in using the LISA cluster.

5

Smoothed ℓ_q-Regularized Regression

G.J.J. van den Burg, P.J.F. Groenen, and A. Alfons

Abstract

An iterative majorization algorithm is presented for the ℓ_q-regularized regression problem that can be used for all $q \in [0,2]$. The algorithm therefore has ridge regression, bridge regression, the lasso, and best subset selection as special cases. The technique of smoothing methods, also known as homotopy methods, is used to tackle this generally noncontinuous nonconvex problem. Through iteratively refining a smooth approximation of the regularized regression problem, local minima of the problem can be avoided. A single unified algorithm is presented based on the idea of graduated nonconvexity. For the proposed method an iterative majorization algorithm is derived and a preliminary convergence analysis is presented, which shows convergence of the intermediate minimizers to the global minimizer. The convergence of the algorithm is further illustrated for the nonconvex case of $q = 0$ using numerical experiments. Results show that the proposed algorithm converges to the global solution for certain algorithm parameters, thereby confirming the feasibility of the algorithm in practice.

5.1 Introduction

Regularized regression has a long history and a wealth of applications in statistics, machine learning, and beyond. The most well-known cases of regularized regression are perhaps ridge regression (Hoerl and Kennard, 1970) and the lasso (Tibshirani, 1996). These methods are special cases of so-called ℓ_q-regularized regression methods, which follow the general form

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} H(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_q^q. \tag{5.1}$$

In this expression $\boldsymbol{\beta}$ denotes the vector of regression coefficients, $H(\boldsymbol{\beta})$ gives the data term of the regression problem, and $\lambda \geq 0$ is a regularization parameter. The norm in the regularization term is defined as

$$\|\boldsymbol{\beta}\|_q^q = \sum_{j=1}^{d} |\beta_j|^q. \tag{5.2}$$

Under a slight abuse of terminology, this function will be referred to as the ℓ_q *norm* in the remainder of this work, unless stated otherwise.

Different values of the parameter q result in different behavior in the obtained solution. In most cases a form of *shrinkage* is applied to the parameters, which results in smaller coefficient values than those obtained by ordinary least squares. For instance, $q = 2$ controls the Euclidean norm of the regression coefficients and thereby applies uniform shrinkage to all coefficients, whereas $q = 1$ applies the Manhattan norm to the coefficients and forces some coefficients to exactly zero while shrinking others. The extreme case where $q = 0$ corresponds to subset selection whereby a limited number of coefficients are included in the model without shrinkage.

An important observation to make about ℓ_q-regularized regression is that the corresponding optimization problem in (5.1) is nonconvex for values of $q < 1$. Furthermore, the ℓ_q norm ceases to be a proper norm if $q = 0$ due to the lack of homogeneity. However, for ease of reference this function will still be called the ℓ_q norm for all values of q throughout this text. As a result of the nonconvexity of the regression problem, most solution methods target either a fixed value of q or a range of values such as $q \in (0,1)$ or $q \in [1,2]$. The case of $q = 0$ has received a particularly large amount of attention due to its computational difficulty (Natarajan, 1995) and its applicability in various domains besides statistics, such as signal processing (e.g. Donoho et al., 2006, Mohimani et al., 2009) and image reconstruction (Trzasko and Manduca, 2009a, Chouzenoux et al., 2013, Nikolova et al., 2008, 2010, among others).

Thus far, no algorithm solves ℓ_q-regularized regression for all $q \in [0,2]$ with a single unified formulation. In this work, the Smooth-q regression algorithm will be presented to solve this problem. Within a single formulation this algorithm can deal with the nonconvex regularized problems whenever $q \in [0,1)$ and with the convex optimization problems for $q \in [1,2]$. The technique used in this algorithm is known as the smoothing, continuation, or homotopy method (Lahaye, 1934, Bertsekas, 1975, among others, see Allgower and Georg, 2012 for an introduction). When applied to nonconvex optimization this idea is also known as graduated nonconvexity (Blake, 1983). The core of the algorithm relies on a smooth and continuous approximation of the ℓ_q penalty, which is iteratively refined until the exact penalty is obtained. At each iteration, an intermediate solution is computed as an improvement on the previous solution. This approach thereby aims to avoid the local minima that are present in the nonconvex problems of $q < 1$.

A significant influence on the development of the Smooth-q algorithm presented here is the work of Mohimani et al. (2009), in which the Smoothed ℓ_0 (SL0) algorithm is introduced. The SL0 algorithm is a method for solving the sparse decomposition problem in compressed sensing, where the ℓ_0 norm of the coefficients in an underdetermined system of linear equations is minimized. Although this problem is not identical to the regularized regression problem, the strategy of the SL0 algorithm can still be used for ℓ_0-regularized regression, as evidenced by the SparseStep algorithm of Van den Burg et al. (2017). In fact, the Smooth-q algorithm presented here is an extension of the SparseStep algorithm to general ℓ_q-regularized regression problems.

A large number of algorithms exist which solve the ℓ_q-regularized regression problem for some values of q, either by using continuation methods or in alternative ways. A full review of these methods is outside the scope of this work, but the most relevant existing alternatives will be mentioned below. Among ℓ_q-regularized regression methods, there is considerable literature available for regularization with $q \in (0,1)$, because for these values of q sparse solutions are obtained. Regularized regression with this penalty is also known as bridge regression (Frank and Friedman, 1993). In Chen et al. (2010) bounds are obtained on the values of the nonzero entries in the solution and error bounds are given for verifying the accuracy of obtained solutions. Finally, a homotopy between ℓ_0 and ℓ_1-regularized regression is presented in Lv and Fan (2009) and applied to sparse recovery.

Smoothing or continuation methods for nonconvex regularized optimization problems have previously been proposed in the literature. In the work of Chen (2012) a convergence analysis is given for a general class of smoothing methods based on the absolute hinge function. Later work by Chen et al. (2013) derived optimality conditions for minimizers of problems with nonconvex penalties, as well as a trust region Newton method for these problems. In related work of Nikolova et al. (2008, 2010) smoothing methods were applied to image restoration through interior point and gradient-based methods, respectively. Furthermore, Lu (2014) proposed an iteratively reweighted minimization method for a general class of ℓ_q-regularized problems with $q \in (0,1)$ and presented an algorithm based on a continuation method for the ℓ_q norm. A similar method was given by Bian and Chen (2013) for a more general class of regularized minimization problems, with a different smoothing approach for a certain class of penalty functions. Finally, the work of Mobahi and Fisher III (2015) gives a theoretical analysis of the general class of Gaussian homotopy methods for nonconvex optimization.

For ℓ_0-regularized regression many solution methods exist. Modern applications of the subset selection approach include the work of Bertsimas et al. (2016), where a method is presented based on mixed integer optimization that solves the best subset selection for problems of hundreds of variables within minutes. Some methods, such as SCAD (Fan and Li, 2001) or MC+ (Zhang, 2010) replace the ℓ_0 penalty by a smooth penalty, which in turn induces sparsity. Other methods, such as the iterative hard thresholding algorithm by Blumensath and Davies (2008) or the iterative soft thresholding algorithm by Marjanovic et al. (2015) target the ℓ_0-regularized regression directly.

Smoothing methods for the ℓ_0 penalty in image processing includes the work of Trzasko and Manduca (2009a) for image reconstruction and that of Chouzenoux et al. (2013), which analyzes convergence properties of iterative majorization algorithms for smoothed penalties. In Soubies et al. (2015) a continuous exact penalty is introduced, which approximates the ℓ_0 norm. In later work, Soubies et al. (2016) analyze smoothing penalties for the ℓ_0 norm theoretically and give conditions on these penalties to preserve the minimizers of the ℓ_0-regularized problem. The SparseStep algorithm (Van den Burg et al., 2017) on which the Smooth-q algorithm is based, uses the Geman-McClure penalty to smooth the ℓ_0 norm (Geman and McClure, 1985). Although the SparseStep algorithm investigates this penalty for ℓ_0-regularized regression, its potential in sparse signal deconvolution had earlier been recognized by Trzasko and Manduca (2009b), De Rooi and Eilers (2011) and Castella and Pesquet (2015). Finally, the SL0 algorithm mentioned above (Mohimani et al., 2009, 2010) and the tuned version by Oxvig et al. (2013) solve the sparse decomposition problem using the smooth ℓ_0 norm as well.

It is thus evident that a significant number of algorithms exist for ℓ_q-regularized regression for specific values or intervals of $q \in [0,2]$ and that the potential of smoothing methods has been realized especially for ℓ_0-regularized regression.

However, no unified method is applicable for *all* values of $q \in [0,2]$. Through an application of the smoothing methodology, this chapter presents the Smooth-q algorithm to solve the entire class of ℓ_q-regularized regression problems within a single formulation. This algorithm is valid for both the nonconvex problems with $q \in [0,1)$ and the convex problems with $q \in [1,2]$.

This work is structured as follows. In Section 5.2 the theory behind the proposed Smooth-q algorithm will be introduced. Moreover, in this section the optimization algorithm will be derived and a preliminary convergence analysis will be given. Convergence of the algorithm is further analyzed in Section 5.3 in one-dimensional and multivariate numerical applications. Section 5.4 concludes.

5.2 Theory

In this section the Smooth-q penalty will be introduced, the optimization algorithm will be derived, and a preliminary convergence analysis will be given.

5.2.1 *Penalty Function*

The main focus of this work is the analysis of the regularized regression problem with the so-called Smooth-q penalty term, given in the following definition.[1]

DEFINITION 5.2.1. *For $\beta \in \mathbb{R}$, $q \in [0,2]$, $p \geq q$, and $\gamma \geq 0$ the Smooth-q penalty is defined as*

$$P_\gamma(\beta) = \frac{|\beta|^p}{|\beta|^{p-q} + \gamma^{p-q}}. \tag{5.3}$$

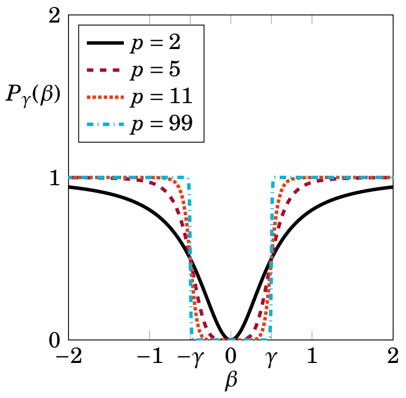
The Smooth-q penalty can be seen as a generalized version of the potential function introduced by Geman and McClure (1985). The special case of this penalty where $p = 2$ and $q = 0$ was previously analyzed by De Rooi and Eilers (2011), Castella and Pesquet (2015) and Van den Burg et al. (2017) among others. The limit case for $\gamma \to 0$ is one of the reasons this penalty is interesting, as

$$\lim_{\gamma \to 0^+} P_\gamma(\beta) = |\beta|^q. \tag{5.4}$$

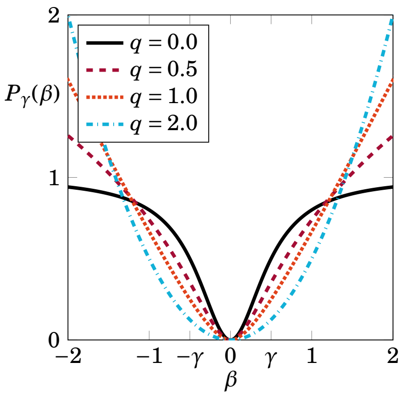
Note that this means that the value of p is irrelevant for the limit behavior of this penalty. Moreover, when $q = 0$ the limit of this penalty becomes

$$\lim_{\gamma \to 0^+} \frac{|\beta|^p}{|\beta|^p + \gamma^p} = \pi[\beta \neq 0], \tag{5.5}$$

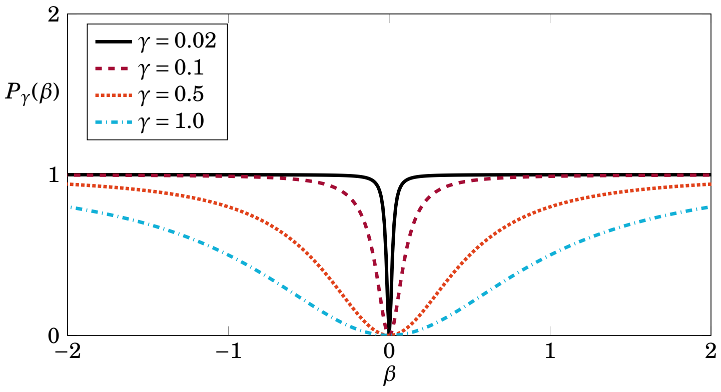
[1]Although the regression problem is the main focus here, the Smooth-q algorithm can easily be extended to other ℓ_q-regularized optimization problems.



(a) $q = 0.0$, $\gamma = 0.5$



(b) $p = 2.0$, $\gamma = 0.5$



(c) $p = 2.0$, $q = 0.0$

FIGURE 5.1 – *Illustrations of the penalty function $P(\beta)$ for varying values of p, q, and γ. Note that all penalties are symmetric around 0.*

where $\pi[I]$ is the indicator function, which equals 1 if I is true and 0 otherwise.

To gain a better understanding of the shape of this penalty, the functional form for several combinations of the parameter p, q, and γ are shown in Figure 5.1. From Figure 5.1(a) it can be observed that increasing p makes the transition area around γ sharper, approaching a discontinuity. Varying q, as shown in Figure 5.1(b), simply changes the limiting function for larger values of $|\beta|$. Finally, by decreasing γ as in Figure 5.1(c) the approximation of the target $|\beta|^q$ becomes increasingly accurate.

The limiting behavior for $\gamma \to 0$ is one of the main motivations for investigating this penalty term, since this corresponds to the ℓ_q-regularized regression problem. Thus, in the limit case $q = 2$ corresponds to ridge regression, $q = 1$ to the lasso,

Algorithm 5.1. Smooth-q Regression

```
1: procedure SmoothQReg($\mathbf{X}$, $\mathbf{y}$, $\lambda$, $q$)
2:     $\boldsymbol{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$
3:     Define a decreasing sequence $(\gamma_k)_{k=1,\ldots,K}$
4:     for $k \in \{1,\ldots,K\}$ do
5:         for $t \in \{1,\ldots,T\}$ do
6:             Update $\boldsymbol{\beta}$ w.r.t. $L_{\gamma_k}(\boldsymbol{\beta})$
7:         end for
8:     end for
9:     return $\boldsymbol{\beta}$
10: end procedure
```

and $q = 0$ to best subset selection. Note that this problem is nonconvex for $q \in [0,1)$. By replacing the nonconvex penalty with the Smooth-q penalty, the nonconvexity in the problem can be introduced slowly by iteratively decreasing γ. This technique is known as Graduated Non-Convexity (Blake, 1983).

5.2.2 *Regularized Regression*

This work focuses on the application of this Smooth-q penalty to the regularized regression problem. Let $\mathscr{D} = \{(\mathbf{x}_i', y_i)\}_{i=1,\ldots,n}$ denote the data for the regression problem, with $\mathbf{x}_i' = (x_{i1},\ldots,x_{id})' \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Moreover, let $\mathbf{X} \in \mathbb{R}^{n\times d}$ denote the data matrix with rows $\mathbf{x}_i'$. Assume that the vector of outcomes $\mathbf{y} \in \mathbb{R}^n$ is centered, so that the intercept term can be ignored. In practice both $\mathbf{X}$ and $\mathbf{y}$ can additionally be normalized by $\sqrt{n}$.

The goal is to solve the ℓ_q-regularized regression problem of the form

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} L_0(\boldsymbol{\beta}) := \left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda \sum_{j=1}^{d} |\beta_j|^q \tag{5.6}$$

where $\lambda > 0$. This problem is approximated with the Smooth-q penalty in the limit for $\gamma \to 0$. Therefore, the Smooth-q regularized regression problem can be formally stated as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \lim_{\gamma\to 0^+} L_\gamma(\boldsymbol{\beta}) := \left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda \sum_{j=1}^{d} P_\gamma(\beta_j). \tag{5.7}$$

The solution approach for this problem is similar to that of other Graduated Non-Convexity approaches, such as those presented in Mohimani et al. (2009, 2010) or Chouzenoux et al. (2013), among others. The general form of the algorithm is given in Algorithm 5.1. Although this algorithm is defined very generally, a number of observations can be made. First, note that $\boldsymbol{\beta}$ is initialized using the ordinary least squares solution, as this corresponds to the case where $\gamma \to \infty$.[2] Next, a decreasing sequence of γ values is defined and for each value of γ_k the inner loop does T steps of an optimization algorithm. The optimization strategy used in the inner loop can be varied, but common choices are gradient descent (Mohimani et al., 2009) or iterative majorization (Chouzenoux et al., 2013). Note that it is not generally necessary to fully minimize the loss function for a given value of γ_k, as this may cause the algorithm to get stuck in local minima. In practice only a few iterations of the inner loop are performed.

To complete the description of the Smooth-q regression algorithm, a few choices have to be made with regards to the optimization algorithm in the inner loop of Algorithm 5.1, the number of steps T, and the sequence of γ values $(\gamma_k)_{k=1,\dots,K}$. As in Mohimani et al. (2009, 2010) a geometric series will be used for the γ_k, such that $\gamma_{k+1} = c \cdot \gamma_k$ for some decay parameter $c \in (0,1)$. For the optimization in the inner loop an iterative majorization algorithm will be derived in the next section.

5.2.3 *Iterative Majorization Algorithm*

Before the optimization algorithm is derived, the shape of the Smooth-q penalty for varying values of p must be investigated further. Figure 5.1(a) illustrates that when $p > 2$ the penalty for $|\beta| < \gamma$ will tend towards zero. Although this may be desirable behavior in specific applications, in the regularized regression problem it is uncommon to have a region around the origin that is not penalized. Moreover, for increasing values of p a discontinuity around $|\beta| = \gamma$ is approached, which may lead to slow convergence of the minimization algorithm. Therefore, the iterative majorization algorithm given here will focus on $p = 2$.

In iterative majorization algorithms (Ortega and Rheinboldt, 1970, De Leeuw, 1994) a majorizing function is derived, which is everywhere larger or equal than the target function to be minimized, but touches at a so-called *supporting point*. Iteratively minimizing the majorizing function will give a path of solutions which converges to a (local) minimum of the target function. In practice the majorizing function has a simple functional form to ensure that minimization can be done quickly. Here a quadratic majorizing function will be used of the form

$$g(\beta, \bar{\beta}) = a\beta^2 - 2b\beta + c \tag{5.8}$$

[2]It is assumed that the OLS solution exists, which may not be unique when $d > n$. Therefore, the general case where $d > n$ is considered outside the scope of the current work and a topic for future research.

where the coefficients a, b, and c depend on the supporting point $\bar{\beta}$.

The derivation starts by constructing a majorizing function for the one-dimensional Smooth-q penalty, since this can easily be generalized to the multivariate case. Since the penalty is everywhere differentiable for $\gamma > 0$, the conditions for the majorizing function can be summarized as

$$P_\gamma\left(\bar{\beta}\right) = g\left(\bar{\beta},\bar{\beta}\right) \tag{5.9}$$

$$P'_\gamma\left(\bar{\beta}\right) = g'\left(\bar{\beta},\bar{\beta}\right) \tag{5.10}$$

$$P_\gamma\left(\beta\right) \geq g\left(\beta,\bar{\beta}\right). \tag{5.11}$$

Since the Smooth-q penalty is a differentiable and even function (i.e. $P_\gamma(\beta) = P_\gamma(-\beta)$ for all β), Theorem 4.5 from De Leeuw and Lange (2009) can be applied, which states that the best quadratic majorizing function of $P_\gamma(\beta)$ is given by

$$g(\beta,\bar{\beta}) = \frac{P'_\gamma(\bar{\beta})}{2\bar{\beta}}(\beta^2 - \bar{\beta}^2) + P_\gamma(\bar{\beta}) \tag{5.12}$$

provided that $P'_\gamma(\beta)/\beta$ is decreasing on $(0,\infty)$. This latter property is readily verified to be the case, because $P'_\gamma(\beta)/\beta$ is continuous and its derivative

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\frac{P'_\gamma(\beta)}{\beta} = \frac{(q-2)\left|\beta\right|^{1-q}\operatorname{sign}(\beta)\left[q\left|\beta\right|^{2-q} - (q-4)\gamma^{2-q}\right]}{\left(\left|\beta\right|^{2-q} + \gamma^{2-q}\right)^3} \tag{5.13}$$

is negative for all $\beta \in (0,\infty)$ since $\operatorname{sign}(\beta) = 1$ and $q \in [0,2]$. The majorizing function is illustrated in Figure 5.2 for two different values of q.

Recall that the loss function of the regression problem with the Smooth-q penalty is given by

$$L_\gamma(\boldsymbol{\beta}) = \left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda\sum_{j=1}^{d} P_\gamma(\beta_j). \tag{5.14}$$

The majorizing function for this loss function can then be constructed from the one-dimensional majorizing functions $g(\beta,\bar{\beta})$ as

$$G(\boldsymbol{\beta},\bar{\boldsymbol{\beta}}) = \left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda\sum_{j=1}^{d} g(\beta_j,\bar{\beta}_j). \tag{5.15}$$

When denoting the coefficients of the majorizing function for β_j respectively by a_j and c_j, it is possible to define a diagonal matrix $\mathbf{A}$ with elements a_j, such that

$$G(\boldsymbol{\beta},\bar{\boldsymbol{\beta}}) = \left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 + \lambda\boldsymbol{\beta}'\mathbf{A}\boldsymbol{\beta} + \lambda\sum_{j=1}^{d} c_j. \tag{5.16}$$

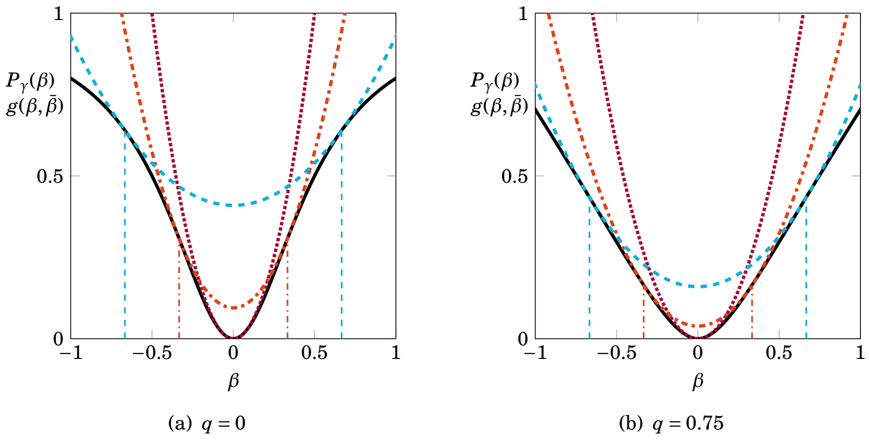


FIGURE 5.2 – *Illustrations of the majorizing function $g(\beta,\bar{\beta})$ for two different values of q, with $\gamma = 0.5$ in both figures. The majorizing function is shown for different values of the supporting point $\bar{\beta}$: the dotted line at $\bar{\beta} = 0$, the dash dotted line at $\bar{\beta} = 1/3$, and the dashed line at $\bar{\beta} = 2/3$. The vertical lines mark these supporting points for the latter two values of $\bar{\beta}$.*

Taking the derivative with respect to $\boldsymbol{\beta}$ and equating this to zero yields

$$\left(\mathbf{X}'\mathbf{X} + \lambda\mathbf{A}\right)\boldsymbol{\beta} = \mathbf{X}'\mathbf{y}. \tag{5.17}$$

This equation defines the update at each step of the inner loop in Algorithm 5.1. Note that $\mathbf{A}$ is dependent on γ. Since $\mathbf{A}$ is a diagonal matrix this expression is remarkably similar to the solution of the ridge regression problem, where $\mathbf{A}$ is simply the identity matrix. In fact, for $q = 2$ the Smooth-q penalty is independent of γ and the above expression reduces exactly to ridge regression.

5.2.4 *Convergence Analysis*

In this section a preliminary convergence analysis is given for the Smooth-q algorithm presented above. It will be shown that the infimum of L_{γ_k} converges to the infimum of L_0 and that if L_0 has a unique minimizer then the minimizers of L_{γ_k} converge to it. This proof is analogous to that presented in Chouzenoux et al. (2013). Where appropriate it will be assumed that the matrix $\mathbf{X}$ is of full rank, such that the OLS solution is unique.

For this analysis the most general form of the penalty will be used,

$$P_\gamma(\boldsymbol{\beta}) = \sum_{j=1}^{d} \frac{|\beta_j|^p}{|\beta_j|^{p-q} + \gamma^{p-q}} \tag{5.18}$$

with $p \geq q$, $p \geq 1$, and $q \in [0,2]$. Moreover, write the target function L_0 as

$$L_0(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 + \lambda Q(\boldsymbol{\beta}) \tag{5.19}$$

where

$$Q(\boldsymbol{\beta}) = \begin{cases} \sum_{j=1}^{d} \pi[\beta_j \neq 0] & \text{if } q = 0 \\ \sum_{j=1}^{d} |\beta_j|^q & \text{if } q \in (0,2]. \end{cases} \tag{5.20}$$

To facilitate the proof of the main convergence result, several properties of the functions L_0 and L_γ will be presented. Let $(\gamma_k)_{k\in\mathbb{N}}$ be a decreasing sequence of positive real numbers converging to 0.

LEMMA 5.2.2. *For every $k \in \mathbb{N}$, $L_{\gamma_{k+1}} \geq L_{\gamma_k}$ if $\gamma_{k+1} \leq \gamma_k$.*

Proof. If $\gamma_{k+1} \leq \gamma_k$ then, $\gamma_{k+1}^{p-q} \leq \gamma_k^{p-q}$ since $p \geq q$. Then,

$$\frac{|\beta_j|^p}{|\beta_j|^{p-q} + \gamma_{k+1}^{p-q}} \geq \frac{|\beta_j|^p}{|\beta_j|^{p-q} + \gamma_k^{p-q}} \tag{5.21}$$

this implies that $P_{\gamma_{k+1}}(\boldsymbol{\beta}) \geq P_{\gamma_k}(\boldsymbol{\beta})$ and therefore $L_{\gamma_{k+1}}(\boldsymbol{\beta}) \geq L_{\gamma_k}(\boldsymbol{\beta})$. □

LEMMA 5.2.3. *For every $k \in \mathbb{N}$, L_{γ_k} is a continuous function.*

Proof. Since the Euclidean norm is known to be continuous, L_{γ_k} is a continuous function as long as the penalty term P_{γ_k} is continuous. This depends on whether the function

$$f(x) = \frac{|x|^p}{|x|^{p-q} + \gamma_k^{p-q}} \tag{5.22}$$

is continuous on $x \in \mathbb{R}$ for all p and q satisfying the conditions given above. Continuity of $f(x)$ at $c \in \mathbb{R}$ means that $\lim_{x\to c} f(x) = f(c)$. Clearly, the crucial point for this function lies at $c = 0$. Note that $f(c)$ is defined $\forall c \in \mathbb{R}$, since $\gamma_k > 0$. Next, $\lim_{x\to 0} f(x)$ exists, as

$$\lim_{x\to 0^+} f(x) = 0 \quad \text{and} \quad \lim_{x\to 0^-} f(x) = 0. \tag{5.23}$$

Finally, $\lim_{x\to 0} f(x) = 0 = f(0)$, so $f(x)$ is continuous and so is L_{γ_k}. □

LEMMA 5.2.4. $\sup_{k\in\mathbb{N}} L_{\gamma_k}$ *equals L_0.*

Proof. Since it was shown above that $L_{\gamma_{k+1}} \geq L_{\gamma_k}$ for $\gamma_{k+1} \leq \gamma_k$, the proof reduces to proving that $P_0(\boldsymbol{\beta}) = Q(\boldsymbol{\beta})$. Hence, for $q = 0$ it should hold that

$$\lim_{\gamma \to 0^+} \frac{|x|^p}{|x|^p + \gamma^p} = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{otherwise} \end{cases} \tag{5.24}$$

and for $q > 0$ it should hold that

$$\lim_{\gamma \to 0^+} \frac{|x|^p}{|x|^{p-q} + \gamma^{p-q}} = |x|^q. \tag{5.25}$$

For the first limit, note that if $x = 0$ it holds that

$$\lim_{\gamma \to 0^+} \frac{0}{0 + \gamma^p} = 0, \tag{5.26}$$

and if $x \neq 0$ it follows

$$\lim_{\gamma \to 0^+} \frac{|x|^p}{|x|^p + \gamma^p} = \frac{|x|^p}{|x|^p} = 1. \tag{5.27}$$

For the second limit,

$$\lim_{\gamma \to 0^+} \frac{|x|^p}{|x|^{p-q} + \gamma^{p-q}} = \lim_{\gamma \to 0^+} \frac{|x|^q}{1 + \left(\frac{\gamma}{|x|}\right)^{p-q}} = \frac{|x|^q}{1} = |x|^q \tag{5.28}$$

since $p \geq q$. □

LEMMA 5.2.5. *There is a lower bound to $L_{\gamma_k}, \forall k \in \mathbb{N}$, denoted by $\underline{L}$.*

Proof. Since $P_{\gamma_k}(\boldsymbol{\beta}) \geq 0$, $\forall k \in \mathbb{N}$ and $\lambda > 0$, it follows that

$$L_{\gamma_k}(\boldsymbol{\beta}) \geq \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 = \underline{L}(\boldsymbol{\beta}) \tag{5.29}$$

is a lower bound. □

LEMMA 5.2.6. *$\underline{L}$ is level-bounded.*

Proof. Recall that $\underline{L} : \mathbb{R}^d \to \mathbb{R}$ is level-bounded if for every $\alpha \in \mathbb{R}$ the sublevel set $\text{lev}_{\leq\alpha} \underline{L}$ is bounded and possibly empty (Rockafellar and Wets, 1998, Definition 1.8).[3] Here, $\text{lev}_{\leq\alpha} \underline{L}$ is defined as

$$\text{lev}_{\leq\alpha} \underline{L} = \left\{\boldsymbol{\beta} \in \mathbb{R}^d : \underline{L}(\boldsymbol{\beta}) \leq \alpha\right\} = \left\{\boldsymbol{\beta} \in \mathbb{R}^d : \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 \leq \alpha\right\}. \tag{5.30}$$

Recall further that it has been assumed that the matrix $\mathbf{X}$ is of full rank, such that the OLS solution $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \left\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\right\|^2$ is unique. Let $\alpha^* = \left\|\mathbf{X}\boldsymbol{\beta}^* - \mathbf{y}\right\|^2$.

[3]A set in a metric space is bounded if it is contained in a ball of finite radius.

Then, if $\alpha < \alpha^*$ the sublevel set $\text{lev}_{\leq\alpha}\underline{L}$ is empty. Next, if $\alpha > \alpha^*$ then the sublevel set is bounded due to the convexity of the norm function. In fact, the radius r of the ball centered at $\boldsymbol{\beta}^*$ which bounds the sublevel set is given by the largest Euclidean distance from $\boldsymbol{\beta}^*$ to any $\boldsymbol{\beta}' \in \text{lev}_{\leq\alpha}$ for which $\left\|\mathbf{X}\boldsymbol{\beta}' - \mathbf{y}\right\|^2 = \alpha$. □

The final lemma shows that L_0 is lower semi-continuous (lsc). Before introducing this lemma, two properties of lsc functions are stated. Recall that a function $f : \mathbb{R}^d \to \mathbb{R}$ is lsc if and only if the set

$$\left\{x \in \mathbb{R}^d : f(x) > \alpha\right\} \tag{5.31}$$

is an open set for all $\alpha \in \mathbb{R}$. Proofs of the following properties can be found in Pedersen (1989, Proposition 1.5.12).

PROPERTY 5.2.7. *If $f : \mathbb{R}^d \to \mathbb{R}$ is lsc then $g : \mathbb{R}^d \to \mathbb{R}$ defined by $g(x) = \lambda f(x)$ is also lsc for $\lambda > 0$.*

PROPERTY 5.2.8. *If $f_1, f_2 : \mathbb{R}^d \to \mathbb{R}$ are lsc functions then $f : \mathbb{R}^d \to \mathbb{R}$ defined by $f = f_1 + f_2$ is also lsc.*

LEMMA 5.2.9. *Let $\pi_j : \mathbb{R}^d \to \mathbb{R}$ be given by $\pi_j(x) = \pi[x_j \neq 0]$ for $x \in \mathbb{R}^d$. Then π_j is lsc.*

Proof. The set given by

$$\mathcal{S} = \left\{x \in \mathbb{R}^d : \pi_j(x) > \alpha\right\} \tag{5.32}$$

must be open for all $\alpha \in \mathbb{R}$ for π_j to be lsc. Note that there are two distinct function values for $\pi_j(x)$, hence there are three distinct sets depending on the value of α:

$$\mathcal{S} = \mathbb{R}^d \qquad \alpha < 0 \tag{5.33}$$

$$\mathcal{S} = \mathbb{R}^d \setminus \{0\} \qquad \alpha \in [0,1) \tag{5.34}$$

$$\mathcal{S} = \emptyset \qquad \alpha \geq 1 \tag{5.35}$$

Each of these sets is open, so π_j is lsc. □

LEMMA 5.2.10. *L_0 is lsc.*

Proof. Recall that L_0 is defined as

$$L_0(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 + \lambda Q(\boldsymbol{\beta}). \tag{5.36}$$

Note that $\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2$ is an lsc function since the complement of the set in (5.31) is closed. Moreover, $\left|\beta_j\right|^q$ is an lsc function for $q > 0$ because it is continuous. Then, by Lemma 5.2.9 it follows that $Q(\boldsymbol{\beta})$ is lsc. Combining this with Property 5.2.7 and 5.2.8 completes the proof. □

Before introducing the main result, three theorems from Rockafellar and Wets (1998) are presented on which the main result is based.

THEOREM 5.2.11. *Let $\{f^\nu\}_{\nu\in\mathbb{N}}$ be a sequence of functions on $\mathbb{R}^d$. If the sequence is nondecreasing ($f^\nu \le f^{\nu+1}$), then* e-lim$_\nu f^\nu$ *exists and equals* $\sup_\nu[\operatorname{cl} f^\nu]$ *(Rockafellar and Wets, 1998, Theorem 7.4(d)).*

THEOREM 5.2.12. *The sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ is eventually level-bounded if there is a level-bounded function g such that eventually $f^\nu \ge g$, or if the sequence of sets $\operatorname{dom} f^\nu$ is eventually bounded (Rockafellar and Wets, 1998, Theorem 7.32(a)).*

THEOREM 5.2.13. *Suppose the sequence $\{f^\nu\}_{\nu\in\mathbb{N}}$ is eventually level-bounded, and $f^\nu \xrightarrow{e} f$ with f^ν and f lower semi-continuous and proper. Then*

$$\inf f^\nu \to \inf f \quad \textit{(finite)} \tag{5.37}$$

while for ν in some index set $N \in \mathcal{N}_\infty$ the sets $\operatorname{argmin} f^\nu$ *are nonempty and form a bounded sequence with*

$$\limsup_\nu(\operatorname{argmin} f^\nu) \subset \operatorname{argmin} f. \tag{5.38}$$

Indeed, for any choice of $\varepsilon^\nu \searrow 0$ and $x^\nu \in \varepsilon^\nu - \operatorname{argmin} f^\nu$, the sequence $\{x^\nu\}_{\nu\in\mathbb{N}}$ is bounded and such that all its cluster points belong to $\operatorname{argmin} f$. *If* $\operatorname{argmin} f$ *consists of a unique point $\tilde{x}$, one must actually have $x^\nu \to \tilde{x}$ (Rockafellar and Wets, 1998, Theorem 7.33).*

These theorems allow for the proof of the main convergence result of Smooth-q regression, which is analogous to Proposition 2.4 from Chouzenoux et al. (2013).

THEOREM 5.2.14. *Let L_γ be defined as above and $(\gamma_k)_{k\in\mathbb{N}}$ be a decreasing sequence of positive real numbers converging to 0. Then,*

(i) $\inf L_{\gamma_k} \to \inf L_0$ *as $k \to \infty$.*

(ii) If $\forall k \in \mathbb{N}$, $\hat{\boldsymbol{\beta}}_k$ is a minimizer of L_{γ_k}, then the sequence $(\hat{\boldsymbol{\beta}}_k)_{k\in\mathbb{N}}$ is bounded and all its cluster points are minimizers of L_0.

(iii) If L_0 has a unique minimizer $\tilde{\boldsymbol{\beta}}$, then $\hat{\boldsymbol{\beta}}_k \to \tilde{\boldsymbol{\beta}}$ as $k \to \infty$.

Proof. After Chouzenoux et al. (2013, Proposition 2.4): By Lemma 5.2.2 the sequence $(L_\gamma)_{k\in\mathbb{N}}$ is a nondecreasing sequence of functions on $\mathbb{R}^d$. Therefore according to Theorem 5.2.11 this sequence epi-converges to $\sup_{k\in\mathbb{N}} L_{\gamma_k}$. This function is equal to L_0 following Lemma 5.2.4.

By Lemma 5.2.5 there is a lower bound to the sequence $(L_{\gamma_k})_{k\in\mathbb{N}}$, given by $\underline{L}$, and $\underline{L}$ is level-bounded by virtue of Lemma 5.2.6. Thus, by Theorem 5.2.12 the sequence $(L_{\gamma_k})_{k\in\mathbb{N}}$ is eventually level-bounded.

Finally, since L_0 is lower semi-continuous by Lemma 5.2.10 and it is proper[4], the proof is completed by the application of Theorem 5.2.13. □

This section is concluded with the following conjecture about the convergence properties of the Smooth-q algorithm presented above. The sequence of γ_k is again defined as $\gamma_{k+1} = c\cdot\gamma_k$ with $c \in (0,1)$ a decay constant.

CONJECTURE 5.2.15. *Given sufficient iterative majorization steps T and a large enough decay constant $c \in (0,1)$, the solution $\hat{\boldsymbol{\beta}}_{c,T}$ found by the Smooth-q regularized regression algorithm comes arbitrarily close to the global minimizer of $L_0(\boldsymbol{\beta})$. In other words, if*

$$\tilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^d} L_0(\boldsymbol{\beta}) \tag{5.39}$$

then for any $\delta > 0$ there exists $c \in (0,1)$ and $T > 0$ such that

$$\|\hat{\boldsymbol{\beta}}_{c,T} - \tilde{\boldsymbol{\beta}}\| < \delta. \tag{5.40}$$

A proof of this conjecture is considered outside the scope of the current work. However, in the next section numerical experiments are presented in support of this statement.

5.3 Numerical Explorations

In this section some numerical experiments are presented that explore the convergence properties of the Smooth-q regression algorithm. The aim is to gain an understanding of the role of the decay parameter $c \in (0,1)$ in the series $\gamma_{k+1} = c\cdot\gamma_k$ and the number of majorization steps T in the optimization algorithm. For simplicity, the experiments are restricted to the case where $q = 0$, but they can easily be generalized to other values of q. It is expected that since the nonconvexity is strongest for $q = 0$, these experiments can give insight in the worst-case convergence properties of the Smooth-q algorithm.

5.3.1 *One-dimensional Experiment*

In the first experiment the behavior of the Smooth-q algorithm around the discontinuity in the ℓ_0 penalty will be investigated. The one-dimensional Smooth-

[4] $L_0 > -\infty$ for all $\boldsymbol{\beta} \in \mathbb{R}^m$ and $L_0 < \infty$ for at least one $\boldsymbol{\beta} \in \mathbb{R}^m$. This clearly holds for L_0.

q regression problem is considered of the form

$$L_\gamma(\beta) = (\beta - u)^2 + P_\gamma(\beta), \tag{5.41}$$

with $u \in \mathbb{R}$. For $\gamma \to 0^+$ and $q = 0$, this problem reduces to the problem

$$L_0(\beta) = (\beta - u)^2 + \pi[\beta \neq 0], \tag{5.42}$$

for which the solution is known to be

$$\tilde{\beta}(u) = \begin{cases} 0 & \text{if } u^2 \leq 1 \\ u & \text{otherwise.} \end{cases} \tag{5.43}$$

To illustrate how the convergence properties of the Smooth-q algorithm depend on c and T the following experiment is performed. Let $\mathscr{U}$ denote a set of 100,001 equally spaced points on the interval $[-1.5, 1.5]$. Then, for a given c and T run the Smooth-q algorithm for each $u \in \mathscr{U}$ and compute solutions $\hat{\beta}_{c,T}(u)$. The average error of the Smooth-q algorithm in this experiment can then be denoted by

$$\mathscr{E}(c, T) = \frac{1}{|\mathscr{U}|} \sum_{u \in \mathscr{U}} \left| \tilde{\beta}(u) - \hat{\beta}_{c,T}(u) \right|. \tag{5.44}$$

Note that there is a limited resolution in this experiment. Cases where $\hat{\beta}_{c,T}(u)$ does not exactly equal $\tilde{\beta}(u)$, will likely occur around the strong discontinuity at $|u| = 1$. Thus, when $\mathscr{E}(c, T) = 0$ for some c and T, it can only be stated with certainty that there is no error for all u outside the regions where $|u| = 1 \pm 3 \cdot 10^{-5}$ (due to the choice of the set $\mathscr{U}$).

Figure 5.3 shows the results of this experiment for different values of c and T. As can be seen, the error indeed decreases for increasing values of c and T, in accordance with the convergence conjecture above. For most combinations of c and T there is a small region around the discontinuity where an error in the β estimates occurs. Furthermore, for smaller values of c it can be seen that the error flattens out with increasing numbers of iterations T. This indicates that for those decay constants an increasing number of majorization iterations will not improve the estimate further, as it has converged to a local minimum of the loss function. Finally, for a large enough value of c and T an error of zero can indeed be found, as expected.

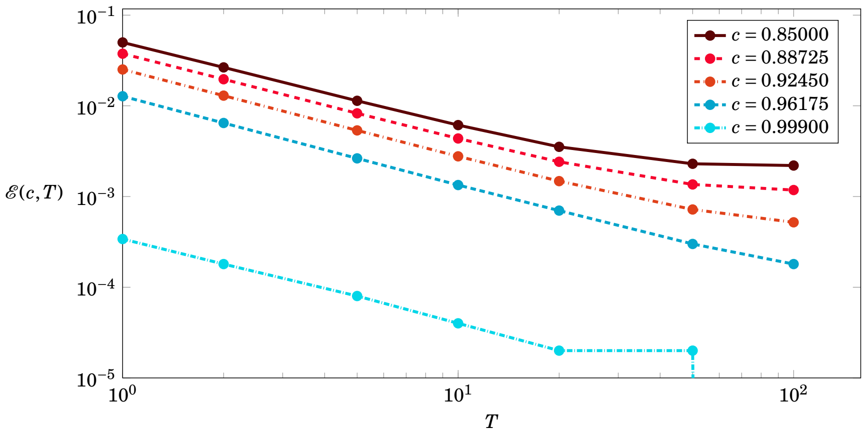


FIGURE 5.3 – *Plot of the average absolute error $\mathscr{E}(c,T)$ for different values of c and T. The figure shows that for larger values of c and T the error decreases, as expected from Conjecture 5.2.15. The result $\mathscr{E}(0.999,100) = 0$ explains the discontinuity in the lowest line.*

5.3.2 *Multidimensional Experiment*

In the multidimensional experiment, the conjecture stated above will be targeted directly. Again the experiment will be restricted to the case where $q = 0$, as the nonconvexity is strongest for this value of q. Moreover, for $q = 0$ it is straightforward to determine the solution $\tilde{\boldsymbol{\beta}}$ through best subset selection. To verify the conjecture a number of possible values of c are chosen as well as a number of values for δ. It will then be shown that for each δ there is a c and a T for which the distance between the solution obtained by the Smooth-q algorithm and that obtained by the brute force algorithm is smaller than δ.

A brief description of the experimental setup follows, further details are given in subsequent paragraphs. First, generate a regression dataset with a true parameter vector $\boldsymbol{\beta}$ which contains r nonzeros. For this dataset, compute the best subset solution $\tilde{\boldsymbol{\beta}}$ of r variables using a brute force algorithm. Second, for a given value of c determine the value of the regularization parameter λ in the Smooth-q algorithm which yields a solution with r nonzeros. Finally, for a given value of δ, determine the smallest value of T for which the distance between $\hat{\boldsymbol{\beta}}_{c,T}$ and $\tilde{\boldsymbol{\beta}}$ is smaller than δ, up to some maximum value of T. Repeat this experiment for several regression datasets.

The regression datasets are generated as follows. First, a data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ is generated with rows $\mathbf{x}_i' \sim \mathcal{N}(\boldsymbol{\mu},\mathbf{I})$ where $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\mu_j \sim U(0,1)$ for all j, and $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix. Next, generate $\boldsymbol{\beta} \in \mathbb{R}^d$ with $\beta_j \sim U(-1,1)$

and set $d - r$ elements of $\boldsymbol{\beta}$ to zero such that it has r nonzero elements. Finally, generate a noise term $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ with $\varepsilon_i \sim \mathcal{N}(0,1)$ and compute $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$.

As mentioned above, the solution $\tilde{\boldsymbol{\beta}}$ can be computed by best subset selection, i.e. by solving the problem

$$\tilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 = r. \tag{5.45}$$

A brute-force approach for this problem can find the optimal solution for small values of r and d within reasonable computation time.

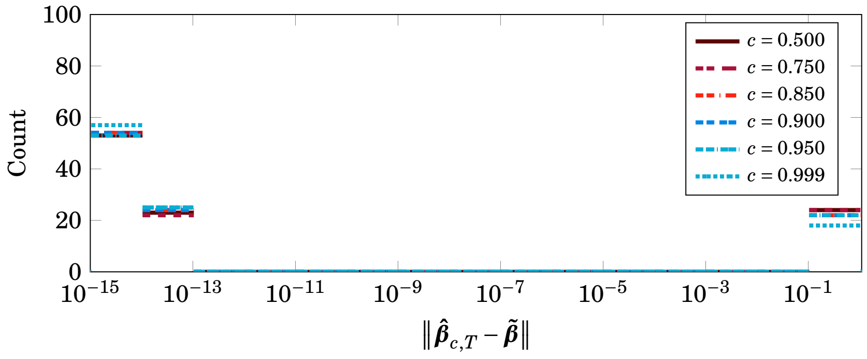
For the Smooth-q algorithm a regularization parameter λ needs to be determined, which yields a solution with r nonzero elements. Finding this λ is done using a binary search algorithm on the range of λ values. This range is recursively partitioned into regions where a distinct number of elements β_j are nonzero. For each possible value of r this technique gives a region where the Smooth-q algorithm finds a solution with r nonzeros.[5]

For each value of c, a value of T is determined which gives an error of at most δ between the Smooth-q and the best subset solutions. This is done by first checking if $T = 1$ yields a close enough solution. If not, the maximum value of $T_{max} = 100$ is used to see if for this large value of T the desired accuracy is reached. If this is the case, the value of T is decreased until the lowest T is found for which the error is still below δ. If T_{max} does not give a solution close enough to $\tilde{\boldsymbol{\beta}}$, it is concluded that the value of c is too small to achieve the desired accuracy.

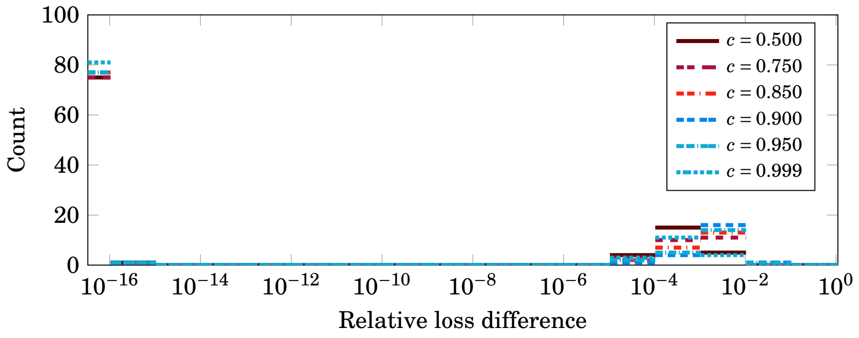
For this experiment, the following parameters are chosen: $n = 500$, $d = 20$, $r = 5$, $\delta \in [10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}]$, and $c \in [0.5, 0.75, 0.85, 0.9, 0.95, 0.999]$. To average the results 100 different regression datasets were generated. The γ sequence used in these experiments follows $\gamma_{k+1} = c \cdot \gamma_k$ with $\gamma_0 = 1/\min\left\{\left|\beta_j^{OLS}\right|\right\}$ and ends when $\gamma_k < 10^{-16}$. Note that the value of γ_0 only needs to be chosen large enough such that the initial Smooth-q solution equals the OLS solution. The suggested starting value seems to work well in practice. Furthermore, the final value of γ_k can also be varied in practice and iterations could also be stopped when the relative change in the parameter values becomes small enough.

Figure 5.4 shows the histograms of the difference in the coefficient estimates and the relative difference in the loss function for the experiments with $\delta = 10^{-12}$. As can be seen from Figure 5.4(a), the difference in the coefficient estimates is below 10^{-13} for the majority of datasets. For these datasets the Smooth-q solution is thus almost exactly equal to the globally optimal solution. The small difference

[5]This binary search algorithm is implemented in the SparseStep R package available on CRAN (Van den Burg et al., 2017)



(a) Histogram of coefficient difference



(b) Histogram of relative loss difference

FIGURE 5.4 – *Histograms of the simulation results of the convergence experiment with* $\delta = 10^{-12}$. *Figure 5.4(a) shows the number of datasets for which the difference in the coefficients estimates lies between* 10^k *and* 10^{k+1} *for* $k = \{-15, \ldots, -1\}$. *Figure 5.4(b) shows a similar histogram for the relative difference in the loss, i.e.* $\left(L\left(\hat{\boldsymbol{\beta}}_{c,T}\right) - L\left(\tilde{\boldsymbol{\beta}}\right)\right)/L\left(\tilde{\boldsymbol{\beta}}\right)$. *For the latter figure the left most interval is between* 0 *and* 10^{-16}.

in accuracy is likely due to rounding errors and machine precision and does not reflect a limitation of the algorithm. This is confirmed in Figure 5.4(b), which shows the relative difference in the loss function between the Smooth-q solution and the globally optimal solution. For the majority of datasets the relative loss difference is between 0 and 10^{-16}.

The results in Figure 5.4 are for $\delta = 10^{-12}$. However, the exact same results are obtained for other values of δ. In other words, the values of δ in the experiment are irrelevant as there does not seem to be a combination of c and T where the obtained solution is below for instance $\delta = 10^{-3}$ but above $\delta = 10^{-12}$. Another result of the experiment is that the value of T does not seem to have any influence on the quality of the solution. For the datasets in this experiment it is observed that if the Smooth-q algorithm finds the optimal solution, it will do so with $T = 1$. Furthermore, if it does not find the solution with $T = 1$ it will not find it with $T = T_{max} = 100$ either. However, the value of c does have an influence on the accuracy of the solution (as confirmed by Figure 5.4). For increasing values of c the number of datasets where the global solution is found increases. For $c \in [0.5, 0.75]$ the global solution is found in 76% of datasets, for $c \in [0.85, 0.9.0.95]$ it is found in 78%, and for $c = 0.999$ it is found in 82% of the datasets.

The datasets where the Smooth-q algorithm does not find the globally optimal solution are of course of interest. For these datasets it might be the case that c needs to be increased further, or the value of the regularization parameter λ needs to be chosen differently to obtain the global solution. Note however that for these datasets the relative difference in the value of the loss function never exceeds about 1%, as indicated in Figure 5.4(b). Further analysis of these datasets is considered a topic for future research. It is finally noted that this experiment would benefit from additional analysis of the convergence for $q \in (0, 2]$, as this does not require determining the value of λ using the binary search algorithm.

5.4 Discussion

The Smooth-q regression algorithm which can be used for ℓ_q-regularized regression with $q \in [0, 2]$ has been presented. This algorithm unifies a number of existing regularized regression methods into a single formulation by employing a smooth continuous approximation of the potentially nonconvex ℓ_q penalty. The algorithm works by iteratively refining this approximation and finding an intermediate solution at each step by iterative majorization.

A preliminary convergence analysis of the Smooth-q algorithm was presented, which shows that the global minimizers of the smoothed loss function converge to the global minimizer of the exact ℓ_q-regularized problem. Moreover, a convergence conjecture was stated, which claims that parameters of the Smooth-q algorithm can be chosen such that the obtained solution comes arbitrarily close to the globally optimal solution. Numerical experiments were conducted which generally support this statement. Future work on the Smooth-q algorithm will focus on theoretical analysis of the convergence conditions. Finally, note that with only minor changes to the loss function the Smooth-q penalty can be used for variations of regularized regression similar to the grouped Lasso (Yuan and Lin, 2006) or the elastic net (Zou and Hastie, 2005). These variations are considered topics for future work as well.

Bibliography

Aizerman, A., Braverman, E.M., and Rozoner, L.I. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2010.

Allgower, E.L. and Georg, K. *Numerical Continuation Methods: An Introduction*, volume 13. Springer Science & Business Media, 2012.

Allwein, E.L., Schapire, R.E., and Singer, Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.

Avi-Itzhak, H. and Diep, T. Arbitrarily tight upper and lower bounds on the Bayesian probability of error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):89–91, 1996.

Ávila Pires, B., Szepesvari, C., and Ghavamzadeh, M. Cost-sensitive multiclass classification risk bounds. In: *Proceedings of the 30th International Conference on Machine Learning*, pp. 1391–1399. 2013.

Bache, K. and Lichman, M. UCI machine learning repository. 2013.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Berisha, V. and Hero, A.O. Empirical non-parametric estimation of the Fisher information. *IEEE Signal Processing Letters*, 22(7):988–992, 2015.

Berisha, V., Wisler, A., Hero, A.O., and Spanias, A. Empirically estimable classification bounds based on a nonparametric divergence measure. *IEEE Transactions on Signal Processing*, 64(3):580–591, 2016.

Bertsekas, D.P. Nondifferentiable optimization via approximation. *Nondifferentiable Optimization*, pp. 1–25, 1975.

Bertsimas, D., King, A., and Mazumder, R. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016.

Bhattacharyya, A. On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics*, pp. 401–406, 1946.

Bian, W. and Chen, X. Worst-case complexity of smoothing quadratic regularization methods for non-Lipschitzian optimization. *SIAM Journal on Optimization*, 23(3):1718–1741, 2013.

Bijleveld, C.C.J.H. and De Leeuw, J. Fitting longitudinal reduced-rank regression models by alternating least squares. *Psychometrika*, 56(3):433–447, 1991.

Blake, A. The least-disturbance principle and weak constraints. *Pattern Recognition Letters*, 1(5):393–399, 1983.

Blumensath, T. and Davies, M.E. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5):629–654, 2008.

Borůvka, O. O jistém problému minimálním. *Práce Moravské Pridovedecké Spolecnosti*, 3:37–58, 1926.

Bottou, L. and Lin, C.J. Support vector machine solvers. In: L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pp. 301–320. MIT Press, Cambridge, MA., 2007.

Bredensteiner, E.J. and Bennett, K.P. Multicategory classification by support vector machines. *Computational Optimization and Applications*, 12(1):53–79, 1999.

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

Castella, M. and Pesquet, J.C. Optimization of a Geman-McClure like criterion for sparse signal deconvolution. In: *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 309–312. 2015.

Cawley, G.C. and Talbot, N.L.C. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

Chang, C.C. and Lin, C.J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

Chapelle, O. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

Chen, X. Smoothing methods for nonsmooth, nonconvex minimization. *Mathematical Programming*, pp. 1–29, 2012.

Chen, X., Niu, L., and Yuan, Y. Optimality conditions and a smoothing trust region Newton method for non-Lipschitz optimization. *SIAM Journal on Optimization*, 23(3):1528–1552, 2013.

Chen, X., Xu, F., and Ye, Y. Lower bound theory of nonzero entries in solutions of $\ell_2 - \ell_p$ minimization. *SIAM Journal on Scientific Computing*, 32(5):2832–2852, 2010.

Chernoff, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pp. 493–507, 1952.

Chouzenoux, E., Jezierska, A., Pesquet, J.C., and Talbot, H. A majorize-minimize subspace approach for ℓ_2–ℓ_0 image regularization. *SIAM Journal on Imaging Sciences*, 6(1):563–591, 2013.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Cox, D.R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

Crammer, K. and Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002a.

Crammer, K. and Singer, Y. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3):201–233, 2002b.

Csiszár, I. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, pp. 146–158, 1975.

Dalcín, L., Paz, R., and Storti, M. MPI for Python. *Journal of Parallel and Distributed Computing*, 65(9):1108–1115, 2005.

De Leeuw, J. Applications of convex analysis to multidimensional scaling. In: J.R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, editors, *Recent Developments in Statistics*, pp. 133–146. North Holland Publishing Company, Amsterdam, 1977.

De Leeuw, J. Convergence of the majorization method for multidimensional scaling. *Journal of Classification*, 5(2):163–180, 1988.

De Leeuw, J. Fitting distances by least squares. Technical Report 130, Los Angeles: Interdivisional Program in Statistics, UCLA, 1993.

De Leeuw, J. Block-relaxation algorithms in statistics. In: H.H. Bock, W. Lenski, and M.M. Richter, editors, *Information Systems and Data Analysis*, pp. 308–324. Springer Berlin Heidelberg, 1994.

De Leeuw, J. and Heiser, W.J. Multidimensional scaling with restrictions on the configuration. *Multivariate Analysis*, 5:501–522, 1980.

De Leeuw, J. and Lange, K. Sharp quadratic majorization in one dimension. *Computational Statistics & Data Analysis*, 53(7):2471–2484, 2009.

De Rooi, J. and Eilers, P. Deconvolution of pulse trains with the L_0 penalty. *Analytica Chimica Acta*, 705(1):218–226, 2011.

Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Dietterich, T.G. and Bakiri, G. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Dogan, U., Glasmachers, T., and Igel, C. Fast training of multi-class support vector machines. Technical Report 03/2011, University of Copenhagen, Faculty of Science, 2011.

Dolan, E.D. and Moré, J.J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

Donoho, D.L. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

Donoho, D.L., Elad, M., and Temlyakov, V.N. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006.

El-Yaniv, R. and Etzion-Rosenberg, N. Hierarchical multiclass decompositions with application to authorship determination. *arXiv preprint arXiv:1010.2102*, 2010.

El-Yaniv, R., Fine, S., and Tishby, N. Agnostic classification of Markovian sequences. In: *Advances in Neural Information Processing Systems 10*, pp. 465–471. 1997.

Fan, J. and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., and Lin, C.J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Frank, E. and Kramer, S. Ensembles of nested dichotomies for multi-class problems. In: *Proceedings of the 21st International Conference on Machine Learning*, pp. 39–46. 2004.

Frank, L.E. and Friedman, J.H. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135, 1993.

Friedman, J.H. and Rafsky, L.C. Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *The Annals of Statistics*, pp. 697–717, 1979.

Friedman, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

Friedman, M. A comparison of alternative tests of significance for the problem of *m* rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

Fukunaga, K. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

Geman, S. and McClure, D.E. Bayesian image analysis: An application to single photon emission tomography. *Proceedings of the American Statistical Association, Statistical Computing Section*, pp. 12–18, 1985.

Groenen, P.J.F. and Heiser, W.J. The tunneling method for global optimization in multidimensional scaling. *Psychometrika*, 61(3):529–550, 1996.

Groenen, P.J.F., Heiser, W.J., and Meulman, J.J. Global optimization in least-squares multidimensional scaling by distance smoothing. *Journal of Classification*, 16(2):225–254, 1999.

Groenen, P.J.F., Nalbantov, G., and Bioch, J.C. Nonlinear support vector machines through iterative majorization and I-splines. In: R. Decker and H.J. Lenz, editors, *Advances in Data Analysis*, pp. 149–161. Springer Berlin Heidelberg, 2007.

Groenen, P.J.F., Nalbantov, G., and Bioch, J.C. SVM-Maj: A majorization approach to linear support vector machines with different hinge errors. *Advances in Data Analysis and Classification*, 2(1):17–43, 2008.

Guermeur, Y. and Monfrini, E. A quadratic loss multi-class SVM for which a radius–margin bound applies. *Informatica*, 22(1):73–96, 2011.

Hardin, J., Garcia, S.R., and Golan, D. A method for generating realistic correlation matrices. *The Annals of Applied Statistics*, 7(3):1733–1762, 2013.

Hardy, G.H., Littlewood, J.E., and Pólya, G. *Inequalities*. Cambridge University Press, 1934.

Hashlamoun, W.A., Varshney, P.K., and Samarasooriya, V.N.S. A tight upper bound on the Bayesian probability of error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):220–224, 1994.

Hastie, T., Tibshirani, R., and Friedman, J.H. *The Elements of Statistical Learning*. Springer, New York, 2nd edition, 2009.

Havel, T.F. An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. *Progress in Biophysics and Molecular Biology*, 56(1):43–78, 1991.

Henze, N. and Penrose, M.D. On the multivariate runs test. *The Annals of Statistics*, pp. 290–298, 1999.

Hill, S.I. and Doucet, A. A framework for kernel-based multi-category classification. *Journal of Artificial Intelligence Research*, 30:525–564, 2007.

Ho, T.K. and Basu, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

Hoerl, A.E. and Kennard, R.W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Holm, S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., and Sundararajan, S. A dual coordinate descent method for large-scale linear SVM. In: *Proceedings of the 25th International Conference on Machine Learning*, pp. 408–415. 2008.

Hsu, C.W. and Lin, C.J. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

Huber, P.J. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

Hubert, L. and Arabie, P. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

Hunter, D.R. and Lange, K. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

Iman, R.L. and Davenport, J.M. Approximations of the critical region of the Friedman statistic. *Communications in Statistics – Theory and Methods*, 9(6):571–595, 1980.

Joachims, T. Training linear SVMs in linear time. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 217–226. 2006.

Kailath, T. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, 1967.

Keerthi, S.S., Sundararajan, S., Chang, K.W., Hsieh, C.J., and Lin, C.J. A sequential dual method for large scale multi-class linear SVMs. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 408–416. 2008.

Kreßel, U.H.G. Pairwise classification and support vector machines. In: B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods*, pp. 255–268. MIT Press, 1999.

Kruskal, J.B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

Kullback, S. and Leibler, R.A. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Lahaye, E. Une méthode de résolution d'une catégorie d'équations transcendantes. *Comptes Rendus de l'Académie des Sciences*, 198:1840–1842, 1934.

Lauer, F. and Guermeur, Y. MSVMpack: A multi-class support vector machine package. *Journal of Machine Learning Research*, 12:2269–2272, 2011.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, Y., Lin, Y., and Wahba, G. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

Lin, C.J., Weng, R.C., and Keerthi, S.S. Trust region Newton method for logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008.

Lin, J. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

Lorena, A.C. and De Carvalho, A.C.P.L.F. Building binary-tree-based multiclass classifiers using separability measures. *Neurocomputing*, 73(16):2837–2845, 2010.

Lorena, A.C., De Carvalho, A.C.P.L.F., and Gama, J.M.P. A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, 30(1):19–37, 2008.

Lu, Y., Guo, H., and Feldkamp, L. Robust neural learning from unbalanced data samples. In: *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 3, pp. 1816–1821. 1998.

Lu, Z. Iterative reweighted minimization methods for ℓ_p regularized unconstrained nonlinear programming. *Mathematical Programming*, 147(1-2):277–307, 2014.

Lv, J. and Fan, Y. A unified approach to model selection and sparse recovery using regularized least squares. *The Annals of Statistics*, pp. 3498–3528, 2009.

March, W.B., Ram, P., and Gray, A.G. Fast euclidean minimum spanning tree: algorithm, analysis, and applications. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 603–612. 2010.

Marjanovic, G., Ulfarsson, M.O., and Hero, A.O. MIST: ℓ_0 sparse linear regression with momentum. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3551–3555. 2015.

Mazumder, R., Friedman, J.H., and Hastie, T. SparseNet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495), 2011.

McCulloch, W.S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

Mobahi, H. and Fisher III, J.W. A theoretical analysis of optimization by Gaussian continuation. In: *AAAI Conference on Artificial Intelligence*, pp. 1205–1211. 2015.

Mohimani, H., Babaie-Zadeh, M., Gorodnitsky, I., and Jutten, C. Sparse recovery using smoothed ℓ^0 (sl0): Convergence analysis. *arXiv preprint arXiv:1001.5073*, 2010.

Mohimani, H., Babaie-Zadeh, M., and Jutten, C. A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm. *IEEE Transactions on Signal Processing*, 57(1):289–301, 2009.

Mroueh, Y., Poggio, T., Rosasco, L., and Slotine, J. Multiclass learning with simplex coding. In: F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pp. 2789–2797. Curran Associates, Inc., 2012.

Natarajan, B.K. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

Nikolova, M., Ng, M.K., and Tam, C.P. Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. *IEEE Transactions on Image Processing*, 19(12):3073–3088, 2010.

Nikolova, M., Ng, M.K., Zhang, S., and Ching, W.K. Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization. *SIAM Journal on Imaging Sciences*, 1(1):2–25, 2008.

Ortega, J.M. and Rheinboldt, W.C. *Iterative Solutions of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.

Oxvig, C.S., Pedersen, P.S., Arildsen, T., and Larsen, T. Surpassing the theoretical 1-norm phase transition in compressive sensing by tuning the smoothed ℓ_0

algorithm. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6019–6023. 2013.

Pearson, K. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

Pedersen, G.K. *Analysis Now*. Springer Verlag, 1989.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., and Cournapeau, D. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peetre, J. and Sparr, G. Interpolation of normed Abelian groups. *Annali di Matematica Pura ed Applicata*, 92(1):217–262, 1972.

Platt, J.C. Fast training of support vector machines using sequential minimal optimization. In: B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods*, pp. 185–208. MIT press, 1999.

Platt, J.C., Cristianini, N., and Shawe-Taylor, J. Large margin DAGs for multiclass classification. In: S.A. Solla, T.K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pp. 547–553. MIT Press, 2000.

Prim, R.C. Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36(6):1389–1401, 1957.

Qiao, X. and Liu, Y. Adaptive weighted learning for unbalanced multicategory classification. *Biometrics*, 65(1):159–168, 2009.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

Rifkin, R. and Klautau, A. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Rockafellar, R.T. *Convex Analysis*. Princeton University Press, 1997.

Rockafellar, R.T. and Wets, R.J.B. *Variational Analysis*. Springer Verlag, 1998.

Rosset, S. and Zhu, J. Piecewise linear regularized solution paths. *The Annals of Statistics*, 35(3):1012–1030, 2007.

Santos, J.M. and Embrechts, M. On the use of the adjusted rand index as a metric for evaluating supervised classification. In: *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, pp. 175–184. 2009.

Schwenker, F. and Palm, G. Tree-structured support vector machines for multi-class pattern recognition. In: *International Workshop on Multiple Classifier Systems*, pp. 409–417. 2001.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

Soubies, E., Blanc-Féraud, L., and Aubert, G. A continuous exact ℓ_0 penalty (CEL0) for least squares regularized problem. *SIAM Journal on Imaging Sciences*, 8(3):1607–1639, 2015.

Soubies, E., Blanc-Féraud, L., and Aubert, G. A unified view of exact continuous penalties for $\ell_2 - \ell_0$ minimization. Technical Report HAL-01267701, INRIA, 2016.

Statnikov, A., Aliferis, C.F., Hardin, D.P., and Guyon, I. *A Gentle Introduction to Support Vector Machines in Biomedicine: Theory and methods*. World Scientific, 2011.

Stoer, M. and Wagner, F. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.

Stone, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147, 1974.

Takahashi, F. and Abe, S. Decision-tree-based multiclass support vector machines. In: *Proceedings of the 9th International Conference on Neural Information Processing*, volume 3, pp. 1418–1422. 2002.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

Tibshirani, R. and Hastie, T. Margin trees for high-dimensional classification. *Journal of Machine Learning Research*, 8(Mar):637–652, 2007.

Trzasko, J. and Manduca, A. Highly undersampled magnetic resonance image reconstruction via homotopic ℓ_0-minimization. *IEEE Transactions on Medical imaging*, 28(1):106–121, 2009a.

Trzasko, J. and Manduca, A. Relaxed conditions for sparse signal recovery with general concave priors. *IEEE Transactions on Signal Processing*, 57(11):4347–4354, 2009b.

Tsutsu, H. and Morikawa, Y. An l_p norm minimization using auxiliary function for compressed sensing. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1. 2012.

Van den Burg, G.J.J. and Groenen, P.J.F. GenSVM: A generalized multiclass support vector machine. *Journal of Machine Learning Research*, 17(225):1–42, 2016.

Van den Burg, G.J.J., Groenen, P.J.F., and Alfons, A. SparseStep: Approximating the counting norm for sparse regularization. *arXiv preprint arXiv:1701.06967*, 2017.

Van den Burg, G.J.J. and Hero, A.O. Fast meta-learning for adaptive hierarchical classifier design. *arXiv preprint arXiv:1711.03512*, 2017.

Van Rossum, G. Python tutorial. Technical Report CS-R9526, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, 1995.

Vapnik, V. *Statistical learning theory*. Wiley, New York, 1998.

Voss, H. and Eckhardt, U. Linear convergence of generalized Weiszfeld's method. *Computing*, 25(3):243–251, 1980.

Vural, V. and Dy, J.G. A hierarchical method for multi-class support vector machines. In: *Proceedings of the 21st International Conference on Machine Learning*, pp. 105–112. 2004.

Wald, A. Foundations of a general theory of sequential decision functions. *Econometrica*, pp. 279–313, 1947.

Weiszfeld, E. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal*, 43:355–386, 1937.

Weston, J. and Watkins, C. Multi-class support vector machines. Technical Report CSD-TR-98-04, University of London, Royal Holloway, Department of Computer Science, 1998.

Whitney, V.K.M. Algorithm 422: minimal spanning tree. *Communications of the ACM*, 15(4):273–274, 1972.

Williams, C.K.I. and Seeger, M. Using the Nyström method to speed up kernel machines. In: T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pp. 682–688. MIT Press, 2001.

Wisler, A., Berisha, V., Wei, D., Ramamurthy, K., and Spanias, A. Empirically-estimable multi-class classification bounds. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2594–2598. 2016.

Yuan, G.X., Chang, K.W., Hsieh, C.J., and Lin, C.J. A comparison of optimization methods and software for large-scale L1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Zhang, C.H. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38:894–942, 2010.

Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Summary

Multiclass classification and regularized regression problems are very common in modern statistics and machine learning applications. On the one hand, multiclass classification is typically concerned with predicting class labels: given observations of objects that belong to certain classes, can we predict to which class a new object belongs? On the other hand, the regression method is often used in economics, social science, medicine, and various other fields to measure how a change in one independent variable influences an observed outcome. In *regularized* regression constraints are placed on the coefficients of the regression model to enforce certain properties in the solution such as sparsity or limited size.

In this dissertation several new algorithms are presented for both multiclass classification and regularized regression. For multiclass classification the GenSVM method is presented. This method extends the binary support vector machine to multiclass classification problems in a way that is both flexible and general, while maintaining competitive predictive performance and training time. In a different chapter accurate estimates of the Bayes error rate are applied to the problem of constructing classification hierarchies: structures in which a multiclass classification problem is decomposed into several binary classification problems. The novelty of the presented method is that accurate estimates of the Bayes error can be used to inform this hierarchy in order to solve the easier problems first before moving on to the more difficult ones.

For regularized regression a new algorithm is presented in two parts: in the first part the problem of ℓ_0-norm regularization is considered and in the second part the entire class of ℓ_q-regularized regression problems is addressed. The difficulty of the ℓ_q-regularized regression problem lies in the fact that for some values of q such problems are hard to solve due to a property known as nonconvexity. In the proposed algorithm a technique known as graduated nonconvexity is used to slowly introduce the nonconvexity in the problem while

iterating towards a solution. The technique is evaluated in the context of ℓ_0-regularized regression using an extensive experimental study. Furthermore, the theoretical properties of the approach are analyzed in the more general ℓ_q-regularized regression problem. The theoretical analysis is supplemented with numerical experiments which show the feasibility of the proposed algorithm for identifying globally optimal solutions.

Samenvatting

Classificatie en geregulariseerde regressie zijn veel voorkomende problemen in de moderne statistiek en machine learning. Bij classificatie is het doel om een klasse te voorspellen: als we observaties hebben van objecten die tot bepaalde klassen behoren, kunnen we dan voorspellen bij welke klasse een nieuw object behoort? De regressie methode wordt in de wetenschap veel gebruikt om te meten hoe een verandering in een onafhankelijke variabele invloed heeft op een geobserveerde uitkomst. In *geregulariseerde* regressie worden beperkingen opgelegd aan de toegestane coëfficiënten van het regressie model om bepaalde eigenschappen van de oplossing te forceren, zoals ijlheid of beperkte grootte van de coëfficiënten.

In dit proefschrift worden verschillende nieuwe algoritmes voor classificatie en geregulariseerde regressie gepresenteerd. Voor classificatie wordt eerst de GenSVM methode gepresenteerd. Deze methode breidt de binaire support vector machine uit naar classificatieproblemen met meer dan twee klassen op een manier die zowel flexibel als algemeen is, zonder daarbij competitieve voorspelkwaliteit of rekentijd op te offeren. In een ander hoofdstuk worden nauwkeurige schattingen van de Bayes fout toegepast op het construeren van classificatiehiërarchieën: structuren waarmee een classificatieprobleem met meer dan twee klassen wordt omgezet naar een reeks binaire problemen. Het vernieuwende in deze methode is dat de nauwkeurige schattingen van de Bayes fout gebruikt kunnen worden om een hiërarchie te construeren zodanig dat de makkelijkere binaire problemen eerst kunnen worden opgelost.

Voor geregulariseerde regressie wordt in twee delen een nieuw algoritme gepresenteerd: in het eerste deel wordt ℓ_0-norm regularisatie bekeken en in het tweede deel wordt de volledige klasse van ℓ_q-norm regularisatie geanalyseerd. De moeilijkheid van ℓ_q-geregulariseerde regressie ligt in het feit dat voor sommige waardes van q dergelijke problemen moeilijk op te lossen zijn door de nonconvexiteit van het probleem. In het voorgestelde algoritme wordt de *stapsgewijze* *nonconvexiteit* techniek gebruikt om de nonconvexiteit in het probleem langzaam te introduceren tijdens de iteraties naar een oplossing. De techniek wordt geëvalueerd met een uitgebreide simulatie studie voor het ℓ_0-geregulariseerde regressie probleem. Vervolgens worden de theoretische eigenschappen geanalyseerd voor het algemene ℓ_q-geregulariseerde probleem. De theoretische analyse wordt aangevuld met numerieke experimenten die laten zien hoe bruikbaar het voorgestelde algoritme is voor het identificeren van de globaal optimale oplossing.

About the Author

Gerrit Jan Johannes (Gertjan) van den Burg grew up in the greenhouse-packed area of Westland, The Netherlands. In 2006 he started the BSc program in Applied Physics at Delft University of Technology. During his studies he joined the pre-master program in Econometrics at the Erasmus University Rotterdam. Between 2009 and 2012 he completed the MSc degree in Applied Physics at Delft University of Technology and the MSc degree in Econometrics and Management Science at the Erasmus University Rotterdam. Gertjan joined ERIM and the Econometric Institute in December 2012.

Gertjan's research focuses on the development of machine learning algorithms, particularly for multiclass classification and regularized regression problems. His research addresses the foundational aspects of machine learning techniques such as loss functions and optimization algorithms, as well as the practical aspects such as algorithm implementation and benchmarking. Gertjan has presented his work at various international conferences and has visited the University of Michigan to collaborate on research. One chapter of his dissertation has been published in the *Journal of Machine Learning Research* whereas another chapter is currently under review at a top machine learning journal.

Gertjan also has significant experience in teaching. During his PhD he was a teaching assistant and thesis supervisor at the BSc and MSc levels and a lecturer for the Matlab module in a BSc programming course. Gertjan pioneered the use of automatically graded programming exercises at the Erasmus School of Economics, for which he received the 2016 Top Educator Award. For more about Gertjan, visit www.gertjanvandenburg.com.

Portfolio

Education

- M.Sc. Econometrics and Management Science, *Erasmus University Rotterdam*, 2010 – 2012.
- M.Sc. Applied Physics, *Delft University of Technology*, 2009 – 2012.
- B.Sc. Applied Physics, *Delft University of Technology*, 2006 – 2009.

Publications

Journal articles

- G.J.J. van den Burg and P.J.F. Groenen. GenSVM: A Generalized Multiclass Support Vector Machine. *Journal of Machine Learning Research*, 17(225):1–42, 2016.

Preprints

- G.J.J. van den Burg, P.J.F. Groenen, and A. Alfons. SparseStep: Approximating the Counting Norm for Sparse Regularization. *Arxiv preprint arXiv:1701.06967*, 2017.
- G.J.J. van den Burg and A.O. Hero. Fast Meta-Learning for Adaptive Hierarchical Classifier Design. *Arxiv preprint arXiv:1711.03512*, 2017.

Selected Working Papers

- G.J.J. van den Burg, P.J.F. Groenen, and A. Alfons. Smoothed ℓ_q-Regularized Regression.

Research Visits

- University of Michigan to visit prof. A.O. Hero, 2017 and 2016.
- Stanford University to visit prof. P.J.F. Groenen, 2014.

Awards and Grants

- Top Educator Award, *Erasmus School of Economics*, 2016.
- Research visit grant, *Erasmus Trustfonds*, 2016.

Selected Conference Presentations

- G.J.J. van den Burg and A.O. Hero. Multiclass Classification and Meta-Learning with Bayes Error Estimates. *Conference of the International Federation of Classification Societies*, Tokyo, 2017.
- G.J.J. van den Burg. Benchmarking Machine Learning Methods on LISA. *SurfSara Super D Event*, Amsterdam, 2016 (invited).
- G.J.J. van den Burg and P.J.F. Groenen. SparseStep: Approximating the Counting Norm for Sparse Regularization. *Conference of the International Federation of Classification Societies*, Bologna, 2015.
- G.J.J. van den Burg and P.J.F. Groenen. An Extended Comparison of Multiclass Support Vector Machines. *21st International Conference on Computational Statistics*, Geneva, Switzerland.
- G.J.J. van den Burg and P.J.F. Groenen. Flexible Multiclass Support Vector Machines. *Conference of the International Federation of Classification Societies*, Tilburg, 2013.

Teaching Activities

At Erasmus University Rotterdam:

- Supervisor for MSc and BSc theses in Econometrics. 2016 – 2017.
- Educational innovation team member, responsible for setting up Autolab for autograding programming exercises, 2016 – 2017.
- Lecturer in Matlab module for undergraduate programming course. 2015 – 2016.

– Teaching Assistant for various undergraduate, graduate, and continuing education courses on programming, data science, econometrics, and mathematics. 2014 – 2016.

PhD Courses

– Reading Group Probabilistic Modeling, *Econometric Institute*.

– Evolutionary Computing, *VU Amsterdam*.

– Convex Analysis for Optimization, *LNMB*.

– Continuous Optimization, *LNMB*.

– Heuristic Methods for Optimization, *LNMB*.

– Summer School on R, *International Association for Statistical Computing*.

– Machine Learning Summer School, *Max Planck Institute for Intelligent Systems*.

– Advanced Statistical Methods, *Erasmus Research Institute in Management*.

– English, *Erasmus Research Institute in Management*.

– Publishing Strategy, *Erasmus Research Institute in Management*.

– Scientific Integrity, *Erasmus Research Institute in Management*.

Languages

Human:

– Dutch: *native*

– English: *fluent*

Computer:

– C, Python, MATLAB: *fluent*

– R, LaTeX: *advanced*

– Java: *intermediate*

The ERIM PhD Series

Alexander, L., *People, Politics, and Innovation: A Process Perspective*. Promotors: Prof. H.G. Barkema & Prof. D.L. van Knippenberg, EPS-2014-331-S&E, http://repub.eur.nl/pub/77209.

Alexiou, A., *Management of Emerging Technologies and the Learning Organization: Lessons from the Cloud and Serious Games Technology*. Promotors: Prof. S.J. Magala, Prof. M.C. Schippers, and Dr. I. Oshri, EPS-2016-404-ORG, http://repub.eur.nl/pub/93818.

Almeida e Santos Nogueira, R.J. de, *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*. Promotors: Prof. U. Kaymak & Prof. J.M.C. Sousa, EPS-2014-310-LIS, http://repub.eur.nl/pub/51560.

Bannouh, K., *Measuring and Forecasting Financial Market Volatility using High-frequency Data*. Promotor: Prof. D.J.C. van Dijk, EPS-2013-273-F&A, http://repub.eur.nl/pub/38240.

Ben-Menahem, S.M., *Strategic Timing and Proactiveness of Organizations*. Promotors: Prof. H.W. Volberda & Prof. F.A.J. van den Bosch, EPS-2013-278-S&E, http://repub.eur.nl/pub/39128.

Benschop, N., *Biases in Project Escalation: Names, frames & construal levels*. Promotors: Prof. K.I.M. Rhode, Prof. H.R. Commandeur, Prof. M. Keil, and Dr. A.L.P. Nuijten, EPS-2015-375-S&E, http://repub.eur.nl/pub/79408.

Berg, W.E. van den, *Understanding Salesforce Behavior using Genetic Association Studies*. Promotor: Prof. W.J.M.I. Verbeke, EPS-2014-311-MKT, http://repub.eur.nl/pub/51440.

Beusichem, H.C. van, *Firms and Financial Markets: Empirical Studies on the Informational Value of Dividends, Governance and Financial Reporting*. Promotors: Prof. A. de Jong & Dr. G. Westerhuis, EPS-2016-378-F&A, http://repub.eur.nl/pub/93079.

Bliek, R. de, *Empirical Studies on the Economic Impact of Trust*. Promotors: Prof. J. Veenman & Prof. Ph.H.B.F. Franses, EPS-2015-324-ORG, http://repub.eur.nl/pub/78159.

Boons, M., *Working Together Alone in the Online Crowd: The Effects of Social Motivations and Individual Knowledge Backgrounds on the Participation and Performance of Members of Online Crowdsourcing Platforms*. Promotors: Prof. H.G. Barkema & Dr. D.A. Stam, EPS-2014-306-S&E, http://repub.eur.nl/pub/50711.

Bouman, P., *Passengers, Crowding and Complexity: Models for Passenger Oriented Public Transport*. Promotors: Prof. L.G. Kroon, Prof. A. Schöbel, and Prof. P.H.M. Vervest, EPS-2017-420-LIS, https://repub.eur.nl/pub/100767.

Brazys, J., *Aggregated Marcoeconomic News and Price Discovery*. Promotor: Prof. W.F.C. Verschoor, EPS-2015-351-F&A, http://repub.eur.nl/pub/78243.

Byington, E., *Exploring Coworker Relationships: Antecedents and Dimensions of Interpersonal Fit, Coworker Satisfaction, and Relational Models*. Promotor: Prof. D.L. van Knippenberg, EPS-2013-292-ORG, http://repub.eur.nl/pub/41508.

Cancurtaran, P., *Essays on Accelerated Product Development*. Promotors: Prof. F. Langerak & Prof. G.H. van Bruggen, EPS-2014-317-MKT, http://repub.eur.nl/pub/76074.

Caron, E.A.M., *Explanation of Exceptional Values in Multi-dimensional Business Databases*. Promotors: Prof. H.A.M. Daniels & Prof. G.W.J. Hendrikse, EPS-2013-296-LIS, http://repub.eur.nl/pub/50005.

Carvalho, L. de, *Knowledge Locations in Cities: Emergence and Development Dynamics*. Promotor: Prof. L. Berg, EPS-2013-274-S&E, http://repub.eur.nl/pub/38449.

Chammas, G., *Portfolio Concentration*. Promotor: Prof. J. Spronk, EPS-2017-410-F&E, https://repub.eur.nl/pub/94975.

Cranenburgh, K.C. van, *Money or Ethics: Multinational corporations and religious organisations operating in an era of corporate responsibility*. Promotors: Prof. L.C.P.M. Meijs, Prof. R.J.M. van Tulder, and Dr. D. Arenas, EPS-2016-385-ORG, http://repub.eur.nl/pub/93104.

Consiglio, I., *Others: Essays on Interpersonal and Consumer Behavior*. Promotor: Prof. S.M.J. van Osselaer, EPS-2016-366-MKT, http://repub.eur.nl/pub/79820.

Cox, R.H.G.M., *To Own, To Finance, and To Insure – Residential Real Estate Revealed*. Promotor: Prof. D. Brounen, EPS-2013-290-F&A, http://repub.eur.nl/pub/40964.

Darnihamedani, P., *Individual Characteristics, Contextual Factors and Entrepreneurial Behavior*. Promotors: Prof. A.R. Thurik & Dr. S.J.A. Hessels, EPS-2016-360-S&E, http://repub.eur.nl/pub/93280.

Dennerlein, T., *Empowering Leadership and Employees' Achievement Motivations: the Role of Self-Efficacy and Goal Orientations in the Empowering Leadership Process*. Promotors: Prof. D.L. van Knippenberg & Dr. J. Dietz, EPS-2017-414-ORG, https://repub.eur.nl/pub/98438.

Deng, W., *Social Capital and Diversification of Cooperatives*. Promotor: Prof. G.W.J. Hendrikse, EPS-2015-341-ORG, http://repub.eur.nl/pub/77449.

Depecik, B.E., *Revitalizing brands and brand: Essays on Brand and Brand Portfolio Management Strategies*. Promotors: Prof. G.H. van Bruggen, Dr. Y.M. van Everdingen, and Dr. M.B. Ataman, EPS-2016-406-MKT, http://repub.eur.nl/pub/93507.

Dollevoet, T.A.B., *Delay Management and Dispatching in Railways*. Promotor: Prof. A.P.M. Wagelmans, EPS-2013-272-LIS, http://repub.eur.nl/pub/38241.

Duyvesteyn, J.G., *Empirical Studies on Sovereign Fixed Income Markets*. Promotors: Prof. P. Verwijmeren & Prof. M.P.E. Martens, EPS-2015-361-F&A, https://repub.eur.nl/pub/79033.

Duursema, H., *Strategic Leadership: Moving Beyond the Leader-Follower Dyad*. Promotor: Prof. R.J.M. van Tulder, EPS-2013-279-ORG, http://repub.eur.nl/pub/39129.

Elemes, A, *Studies on Determinants and Consequences of Financial Reporting Quality*. Promotor: Prof. E. Peek, EPS-2015-354-F&A, https://repub.eur.nl/pub/79037.

Ellen, S. ter, *Measurement, Dynamics, and Implications of Heterogeneous Beliefs in Financial Markets*. Promotor: Prof. W.F.C. Verschoor, EPS-2015-343-F&A, http://repub.eur.nl/pub/78191.

Erlemann, C., *Gender and Leadership Aspiration: The Impact of the Organizational Environment*. Promotor: Prof. D.L. van Knippenberg, EPS-2016-376-ORG, http://repub.eur.nl/pub/79409.

Eskenazi, P.I., *The Accountable Animal*. Promotor: Prof. F.G.H. Hartmann, EPS-2015-355-F&A, http://repub.eur.nl/pub/78300.

Evangelidis, I., *Preference Construction under Prominence*. Promotor: Prof. S.M.J. van Osselaer, EPS-2015-340-MKT, http://repub.eur.nl/pub/78202.

Faber, N., *Structuring Warehouse Management*. Promotors: Prof. M.B.M. de Koster & Prof. A. Smidts, EPS-2015-336-LIS, http://repub.eur.nl/pub/78603.

Feng, Y., *The Effectiveness of Corporate Governance Mechanisms and Leadership Structure: Impacts on strategic change and firm performance*. Promotors: Prof. F.A.J. van den Bosch, Prof. H.W. Volberda, and Dr. J.S. Sidhu, EPS-2017-389-S&E, https://repub.eur.nl/pub/98470.

Fernald, K., *The Waves of Biotechnological Innovation in Medicine: Interfirm Cooperation Effects and a Venture Capital Perspective*. Promotors: Prof. E. Claassen, Prof. H.P.G. Pennings, and Prof. H.R. Commandeur, EPS-2015-371-S&E, http://hdl.handle.net/1765/79120.

Fisch, C.O., *Patents and trademarks: Motivations, antecedents, and value in industrialized and emerging markets*. Promotors: Prof. J.H. Block, Prof. H.P.G. Pennings, and Prof. A.R. Thurik, EPS-2016-397-S&E, http://repub.eur.nl/pub/94036.

Fliers, P.T., *Essays on Financing and Performance: The role of firms, banks and board*. Promotors: Prof. A. de Jong & Prof. P.G.J. Roosenboom, EPS-2016-388-F&A, http://repub.eur.nl/pub/93019.

Fourne, S.P., *Managing Organizational Tensions: A Multi-Level Perspective on Exploration, Exploitation and Ambidexterity*. Promotors: Prof. J.J.P. Jansen & Prof. S.J. Magala, EPS-2014-318-S&E, http://repub.eur.nl/pub/76075.

Gaast, J.P. van der, *Stochastic Models for Order Picking Systems*. Promotors: Prof. M.B.M de Koster & Prof. I.J.B.F. Adan, EPS-2016-398-LIS, http://repub.eur.nl/pub/93222.

Giurge, L.M., *A Test of Time; A temporal and dynamic approach to power and ethics*. Promotors: Prof. M.H. van Dijke & Prof. D. De Cremer, EPS-2017-412-ORG, https://repub.eur.nl/pub/98451.

Glorie, K.M., *Clearing Barter Exchange Markets: Kidney Exchange and Beyond*. Promotors: Prof. A.P.M. Wagelmans & Prof. J.J. van de Klundert, EPS-2014-329-LIS, http://repub.eur.nl/pub/77183.

Hekimoglu, M., *Spare Parts Management of Aging Capital Products*. Promotor: Prof. R. Dekker, EPS-2015-368-LIS, http://repub.eur.nl/pub/79092.

Heyde Fernandes, D. von der, *The Functions and Dysfunctions of Reminders*. Promotor: Prof. S.M.J. van Osselaer, EPS-2013-295-MKT, http://repub.eur.nl/pub/41514.

Hogenboom, A.C., *Sentiment Analysis of Text Guided by Semantics and Structure*. Promotors: Prof. U. Kaymak & Prof. F.M.G. de Jong, EPS-2015-369-LIS, http://repub.eur.nl/pub/79034.

Hogenboom, F.P., *Automated Detection of Financial Events in News Text*. Promotors: Prof. U. Kaymak & Prof. F.M.G. de Jong, EPS-2014-326-LIS, http://repub.eur.nl/pub/77237.

Hollen, R.M.A., *Exploratory Studies into Strategies to Enhance Innovation-Driven International Competitiveness in a Port Context: Toward Ambidextrous Ports*. Promotors: Prof. F.A.J. van den Bosch & Prof. H.W.Volberda, EPS-2015-372-S&E, http://repub.eur.nl/pub/78881.

Hout, D.H. van, *Measuring Meaningful Differences: Sensory Testing Based Decision Making in an Industrial Context; Applications of Signal Detection Theory and Thurstonian Modelling*. Promotors: Prof. P.J.F. Groenen & Prof. G.B. Dijksterhuis, EPS-2014-304-MKT, http://repub.eur.nl/pub/50387.

Houwelingen, G.G. van, *Something To Rely On*. Promotors: Prof. D. de Cremer & Prof. M.H. van Dijke, EPS-2014-335-ORG, http://repub.eur.nl/pub/77320.

Hurk, E. van der, *Passengers, Information, and Disruptions*. Promotors: Prof. L.G. Kroon & Prof. P.H.M. Vervest, EPS-2015-345-LIS, http://repub.eur.nl/pub/78275.

Iseger, P. den, *Fourier and Laplace Transform Inversion with Applications in Finance*. Promotor: Prof. R. Dekker, EPS-2014-322-LIS, http://repub.eur.nl/pub/76954.

Jaarsveld, W.L. van, *Maintenance Centered Service Parts Inventory Control*. Promotor: Prof. R. Dekker, EPS-2013-288-LIS, http://repub.eur.nl/pub/39933.

Khanagha, S., *Dynamic Capabilities for Managing Emerging Technologies*. Promotor: Prof. H.W. Volberda, EPS-2014-339-S&E, http://repub.eur.nl/pub/77319.

Khattab, J., *Make Minorities Great Again: a contribution to workplace equity by identifying and addressing constraints and privileges*. Promotors: Prof. D.L. van Knippenberg & Dr. A. Nederveen Pieterse, EPS-2017-421-ORG, https://repub.eur.nl/pub/99311.

Kil, J., *Acquisitions Through a Behavioral and Real Options Lens*. Promotor: Prof. H.T.J. Smit, EPS-2013-298-F&A, http://repub.eur.nl/pub/50142.

Klooster, E. van't, *Travel to Learn: the Influence of Cultural Distance on Competence Development in Educational Travel*. Promotors: Prof. F.M. Go & Prof. P.J. van Baalen, EPS-2014-312-MKT, http://repub.eur.nl/pub/51462.

Li, D., *Supply Chain Contracting for After-sales Service and Product Support*. Promotor: Prof. M.B.M. de Koster, EPS-2015-347-LIS, http://repub.eur.nl/pub/78526.

Li, Z., *Irrationality: What, Why and How*. Promotors: Prof. H. Bleichrodt, Prof. P.P. Wakker, and Prof. K.I.M. Rohde, EPS-2014-338-MKT, http://repub.eur.nl/pub/77205.

Liu, N., *Behavioral Biases in Interpersonal Contexts*. Promotors: Prof. A. Baillon & Prof. H. Bleichrodt, EPS-2017-408-MKT, https://repub.eur.nl/pub/95487.

Liang, Q.X., *Governance, CEO Identity, and Quality Provision of Farmer Cooperatives*. Promotor: Prof. G.W.J. Hendrikse, EPS-2013-281-ORG, http://repub.eur.nl/pub/39253.

Liket, K., *Why 'Doing Good' is not Good Enough: Essays on Social Impact Measurement*. Promotors: Prof. H.R. Commandeur & Dr. K.E.H. Maas, EPS-2014-307-STR, http://repub.eur.nl/pub/51130.

Loos, M.J.H.M. van der, *Molecular Genetics and Hormones: New Frontiers in Entrepreneurship Research*. Promotors: Prof. A.R. Thurik, Prof. P.J.F. Groenen, and Prof. A. Hofman, EPS-2013-287-S&E, http://repub.eur.nl/pub/40081.

Lu, Y., *Data-Driven Decision Making in Auction Markets*. Promotors: Prof. H.W.G.M. van Heck & Prof. W. Ketter, EPS-2014-314-LIS, http://repub.eur.nl/pub/51543.

Ma, Y., *The Use of Advanced Transportation Monitoring Data for Official Statistics*. Promotors: Prof. L.G. Kroon & Dr. J. van Dalen, EPS-2016-391-LIS, http://repub.eur.nl/pub/80174.

Manders, B., *Implementation and Impact of ISO 9001*. Promotor: Prof. K. Blind, EPS-2014-337-LIS, http://repub.eur.nl/pub/77412.

Mell, J.N., *Connecting Minds: On The Role of Metaknowledge in Knowledge Coordination*. Promotor: Prof. D.L. van Knippenberg, EPS-2015-359-ORG, http://hdl.handle.net/1765/78951.

Meulen,van der, D., *The Distance Dilemma: the effect of flexible working practices on performance in the digital workplace*. Promotors: Prof. H.W.G.M. van Heck & Prof. P.J. van Baalen, EPS-2016-403-LIS, http://repub.eur.nl/pub/94033.

Micheli, M.R., *Business Model Innovation: A Journey across Managers' Attention and Inter-Organizational Networks*. Promotor: Prof. J.J.P. Jansen, EPS-2015-344-S&E, http://repub.eur.nl/pub/78241.

Pozharliev, R., *Social Neuromarketing: The role of social context in measuring advertising effectiveness*. Promotors: Prof. W.J.M.I. Verbeke & Prof. J.W. van Strien, EPS-2017-402-MKT, https://repub.eur.nl/pub/95528.

Pronker, E.S., *Innovation Paradox in Vaccine Target Selection*. Promotors: Prof. H.J.H.M. Claassen, and Prof. H.R. Commandeur, EPS-2013-282-S&E, http://repub.eur.nl/pub/39654.

Protzner, S., *Mind the gap between demand and supply: A behavioral perspective on demand forecasting*. Promotors: Prof. S.L. van de Velde & Dr. L. Rook, EPS-2015-364-LIS, http://repub.eur.nl/pub/79355.

Pruijssers, J.K., *An Organizational Perspective on Auditor Conduct*. Promotors: Prof. J. van Oosterhout & Prof. P.P.M.A.R. Heugens, EPS-2015-342-S&E, http://repub.eur.nl/pub/78192.

Retel Helmrich, M.J., *Green Lot-Sizing*. Promotor: Prof. A.P.M. Wagelmans, EPS-2013-291-LIS, http://repub.eur.nl/pub/41330.

Rietdijk, W.J.R., *The Use of Cognitive Factors for Explaining Entrepreneurship*. Promotors: Prof. A.R. Thurik & Prof. I.H.A. Franken, EPS-2015-356-S&E, http://repub.eur.nl/pub/79817.

Rietveld, N., *Essays on the Intersection of Economics and Biology*. Promotors: Prof. A.R. Thurik, Prof. Ph.D. Koellinger, Prof. P.J.F. Groenen, and Prof. A. Hofman, EPS-2014-320-S&E, http://repub.eur.nl/pub/76907.

Rösch, D., *Market Efficiency and Liquidity*. Promotor: Prof. M.A. van Dijk, EPS-2015-353-F&A, http://repub.eur.nl/pub/79121.

Roza, L., *Employee Engagement in Corporate Social Responsibility: A collection of essays*. Promotor: Prof. L.C.P.M. Meijs, EPS-2016-396-ORG, http://repub.eur.nl/pub/93254.

Rubbaniy, G., *Investment Behaviour of Institutional Investors*. Promotor: Prof. W.F.C. Verschoor, EPS-2013-284-F&A, http://repub.eur.nl/pub/40068.

Schoonees, P., *Methods for Modelling Response Styles*. Promotor: Prof. P.J.F. Groenen, EPS-2015-348-MKT, http://repub.eur.nl/pub/79327.

Schouten, M.E., *The Ups and Downs of Hierarchy: the causes and consequences of hierarchy struggles and positional loss*. Promotors; Prof. D.L. van Knippenberg & Dr. L.L. Greer, EPS-2016-386-ORG, http://repub.eur.nl/pub/80059.

Shahzad, K., *Credit Rating Agencies, Financial Regulations and the Capital Markets*. Promotor: Prof. G.M.H. Mertens, EPS-2013-283-F&A, http://repub.eur.nl/pub/39655.

Smit, J., *Unlocking Business Model Innovation: A look through the keyhole at the inner workings of Business Model Innovation*. Promotor: Prof. H.G. Barkema, EPS-2016-399-S&E, http://repub.eur.nl/pub/93211.

Sousa, M.J.C. de, *Servant Leadership to the Test: New Perspectives and Insights*. Promotors: Prof. D.L. van Knippenberg & Dr. D. van Dierendonck, EPS-2014-313-ORG, http://repub.eur.nl/pub/51537.

Spliet, R., *Vehicle Routing with Uncertain Demand*. Promotor: Prof. R. Dekker, EPS-2013-293-LIS, http://repub.eur.nl/pub/41513.

Staadt, J.L., *Leading Public Housing Organisation in a Problematic Situation: A Critical Soft Systems Methodology Approach*. Promotor: Prof. S.J. Magala, EPS-2014-308-ORG, http://repub.eur.nl/pub/50712.

Stallen, M., *Social Context Effects on Decision-Making: A Neurobiological Approach*. Promotor: Prof. A. Smidts, EPS-2013-285-MKT, http://repub.eur.nl/pub/39931.

Szatmari, B., *We are (all) the champions: The effect of status in the implementation of innovations*. Promotors: Prof. J.C.M. van den Ende & Dr. D. Deichmann, EPS-2016-401-LIS, http://repub.eur.nl/pub/94633.

Tarakci, M., *Behavioral Strategy: Strategic Consensus, Power and Networks*. Promotors: Prof. D.L. van Knippenberg & Prof. P.J.F. Groenen, EPS-2013-280-ORG, http://repub.eur.nl/pub/39130.

Tuijl, E. van, *Upgrading across Organisational and Geographical Configurations*. Promotor: Prof. L. van den Berg, EPS-2015-349-S&E, http://repub.eur.nl/pub/78224.

Tuncdogan, A., *Decision Making and Behavioral Strategy: The Role of Regulatory Focus in Corporate Innovation Processes*. Promotors: Prof. F.A.J. van den Bosch, Prof. H.W. Volberda, and Prof. T.J.M. Mom, EPS-2014-334-S&E, http://repub.eur.nl/pub/76978.

Uijl, S. den, *The Emergence of De-facto Standards*. Promotor: Prof. K. Blind, EPS-2014-328-LIS, http://repub.eur.nl/pub/77382.

Vagias, D., *Liquidity, Investors and International Capital Markets*. Promotor: Prof. M.A. van Dijk, EPS-2013-294-F&A, http://repub.eur.nl/pub/41511.

Valogianni, K., *Sustainable Electric Vehicle Management using Coordinated Machine Learning*. Promotors: Prof. H.W.G.M. van Heck & Prof. W. Ketter, EPS-2016-387-LIS, http://repub.eur.nl/pub/93018.

Vandic, D., *Intelligent Information Systems for Web Product Search*. Promotors: Prof. U. Kaymak & Dr. F. Frasincar, EPS-2017-405-LIS, https://repub.eur.nl/pub/95490.

Veelenturf, L.P., *Disruption Management in Passenger Railways: Models for Timetable, Rolling Stock and Crew Rescheduling*. Promotor: Prof. L.G. Kroon, EPS-2014-327-LIS, http://repub.eur.nl/pub/77155.

Venus, M., *Demystifying Visionary Leadership: In search of the essence of effective vision communication*. Promotor: Prof. D.L. van Knippenberg, EPS-2013-289-ORG, http://repub.eur.nl/pub/40079.

Vermeer, W., *Propagation in Networks:The impact of information processing at the actor level on system-wide propagation dynamics*. Promotor: Prof. P.H.M. Vervest, EPS-2015-373-LIS, http://repub.eur.nl/pub/79325.

Versluis, I., *Prevention of the Portion Size Effect*. Promotors: Prof. Ph.H.B.F. Franses & Dr. E.K. Papies, EPS-2016-382-MKT, http://repub.eur.nl/pub/79880.

Vishwanathan, P., *Governing for Stakeholders: How Organizations May Create or Destroy Value for their Stakeholders*. Promotors: Prof. J. van Oosterhout & Prof. L.C.P.M. Meijs, EPS-2016-377-ORG, http://repub.eur.nl/pub/93016.

Visser, V.A., *Leader Affect and Leadership Effectiveness: How leader affective displays influence follower outcomes*. Promotor: Prof. D.L. van Knippenberg, EPS-2013-286-ORG, http://repub.eur.nl/pub/40076.

Vlaming, R. de., *Linear Mixed Models in Statistical Genetics*. Promotors: Prof. A.R. Thurik, Prof. P.J.F. Groenen, and Prof. Ph.D. Koellinger, EPS-2017-416-S&E, https://repub.eur.nl/pub/100428.

Vries, J. de, *Behavioral Operations in Logistics*. Promotors: Prof. M.B.M de Koster & Prof. D.A. Stam, EPS-2015-374-LIS, http://repub.eur.nl/pub/79705.

Wagenaar, J.C., *Practice Oriented Algorithmic Disruption Management in Passenger Railways*. Promotors: Prof. L.G. Kroon & Prof. A.P.M. Wagelmans, EPS-2016-390-LIS, http://repub.eur.nl/pub/93177.

Wang, P., *Innovations, status, and networks*. Promotors: Prof. J.J.P. Jansen & Dr. V.J.A. van de Vrande, EPS-2016-381-S&E, http://repub.eur.nl/pub/93176.

Multiclass classification and regularized regression problems are very common in modern statistical and machine learning applications. On the one hand, multiclass classification problems require the prediction of class labels: given observations of objects that belong to certain classes, can we predict to which class a new object belongs? On the other hand, the regularized regression problem is a variation of the common regression problem, which measures how changes in independent variables influence an observed outcome. In regularized regression, constraints are placed on the coefficients of the regression model to enforce certain properties in the solution, such as sparsity or limited size.

In this dissertation several new algorithms are presented for both multiclass classification and regularized regression problems. For multiclass classification the GenSVM method is presented. This method extends the binary support vector machine to multiclass classification problems in a way that is both flexible and general, while maintaining competitive performance and training time. In a different chapter, accurate estimates of the Bayes error are applied to both meta-learning and the construction of so-called classification hierarchies: structures in which a multiclass classification problem is decomposed into several binary classification problems.

For regularized regression problems a new algorithm is presented in two parts: first for the sparse regression problem and second as a general algorithm for regularized regression where the regularization function is a measure of the size of the coefficients. In the proposed algorithm graduated nonconvexity is used to slowly introduce the nonconvexity in the problem while iterating towards a solution. The empirical performance and theoretical convergence properties of the algorithm are analyzed with numerical experiments that demonstrate the ability for the algorithm to obtain globally optimal solutions.

ERiM

ERIM PhD Series
Research in Management

Erasmus University Rotterdam (EUR)
Erasmus Research Institute of Management
Mandeville (T) Building
Burgemeester Oudlaan 50
3062 PA Rotterdam, The Netherlands

P.O. Box 1738
3000 DR Rotterdam, The Netherlands
T +31 10 408 1182
E info@erim.eur.nl
W www.erim.eur.nl