Support Vector Machines: Proposals For Dealing With Misclassifications

by

Wim van Duuren (SNR: 2039796)

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor in Econometrics and Operations Research

Tilburg School of Economics and Management
Tilburg University

Supervised by: Prof. Dr. Juan Vera

Date: January 17, 2025

Abstract

A support vector machine is a binary classifier that penalizes misclassifications. The performance of the model decreases as the data contains more noise and outliers. This thesis proposes a method based on bootstrapping using the L_1 SVM to deal more effectively with misclassifications. Furthermore, a capped L_1 cost function is proposed to deal with outliers. Penalty functions from the field of feature selection, such as L_0 and SCAD, are used to provide new misclassification handling cost functions. We investigate if, with these alterations, information about the support vectors is obtained and the kernel trick can be applied. We discover that the bootstrapping method outperforms the L_1 and L_2 SVM when noise and outliers are present in the data. When outliers become huge, the capped L_1 SVM achieves higher scores. Feature selection functions do not perform well when used as cost functions dealing with misclassifications. The recommendation is to not use feature selection functions without algorithms that lower the runtime of the models. On the other hand, we advise to use bootstrapping when encountering data with noise and outliers and to use the capped L_1 SVM when the outliers present are very large.

Contents

Chapter 1

Introduction

Support vector machines (SVM) aim to solve the binary classification problem: given sample data with points belonging to two different classes, decide which class a new point belongs to. An SVM solves the problem by constructing a hyperplane based on the sample data that separates the classes. The hyperplane is called the separation boundary. An example of a binary classification problem is separating cupboards from chairs (Murty and Raghava, 2016). Cupboards and chairs are the two classes, and there is information about their height and weight. The objective is to decide if an unclassified object is a cupboard or a chair whilst only knowing the height and weight of this entity. Based on a set of training data, with classes known for every point, a boundary that separates these classes is calculated. If the new point is on one side of the separation boundary, it is classified as a chair; if it is on the other side, it is classified as a cupboard. Support vector machines can solve this problem when the data is nice enough (Kecman, 2004). It becomes complicated when data is misclassified. A cupboard could, for example, be wrongly labeled as a chair in the training data or the training data could contain outliers, such as a cupboard that weighs much. Ideas on how to deal with noise and outliers in a binary classification problem exist, but unfortunately, they do not work in all cases. This thesis tries to find better ways to deal with misclassifications.

Vladimir Vapnik and Alexey Chervonenkis invented the first support vector machine in 1964. They published two papers, Vapnik and Chervonenkis (1964a) and Vapnik and Chervonenkis (1964b), about the SVM in a linear hard margin setting. The excellent thing about an SVM is that a small subset of points can be identified where the separating boundary can solely be based upon. This is preferable because computation becomes faster and the generalization error is lower (Boser et al., 1992). These points are called support vectors. From the support vectors, weights for the features are calculated. The separating boundary that solves the classification problem is derived with this weight vector and a bias term.

The first SVM created a linear boundary. Boser et al. (1992) found a way to implement the SVM in a nonlinear setting. They wrote the model in dual

form and applied the kernel trick to map the problem efficiently to a higher dimension and solve the linear problem in that dimension before mapping it back to the original dimension. Their SVM was not able to deal with misclassifications. Cortes and Vapnik (1995) invented the soft margin SVM. The model became more flexible: points from different classes did not need to be on different sides of the boundary. This allowed for misclassifications in the problem.

Mangasarian and Bradley (1998) proposed to select a subset of important features from the data by using a penalty function that sets the weight of some features to zero. This reduced the problem's dimensionality even more, lowering the generalization error further (Neumann et al., 2005). In this thesis, feature selection itself is not used, but only concepts developed in this field. Ideas are adapted from Mangasarian and Bradley (1998), Neumann et al. (2005), Zou and Hastie (2005), Zhang et al. (2006) and Becker et al. (2011) to the soft margin support vector machine from Cortes and Vapnik (1995).

Support vector machines have applications in many fields. For example, in medical imaging, medical diagnoses, image interpolation, pattern recognition, geology, and time series (Boyle, 2011). In Yu et al. (2010), the authors use the nonlinear soft margin SVM to predict if a U.S. citizen has diabetes using family history, age, race, ethnicity, weight, height, waist circumference, BMI and hypertension as variables. On the application of the SVM in pattern recognition, Borah and Gupta (2017) mentions that the model can classify 100 objects after being trained with 18 pictures for each object. They note similar performance when applied to facial recognition. Most of these applications use the soft margin SVM allowing for misclassifications. Developing more effective methods to handle misclassifications could enhance these applications, showing the necessity for research into possible improvements in this area.

In Chapter 2, the reader is provided with the basic concepts of this thesis. The notation and the structure of the data are presented. The chapter then discusses the basic concept of a support vector machine. It shows the technique to obtain information about the supporting vectors and to use a support vector machine in a nonlinear setting. Lastly, we examine the literature dealing with misclassifications and feature selection. The datasets are generated in Chapter 3. In Chapter 4, the benchmark models are defined, consisting of models known from the literature, and a method based on bootstrapping, a model using a capped L_1 cost function and models using penalty functions from the field of feature selection as cost functions are proposed. Chapter 5 contains the results from the tests of the models on the datasets. Here, runtime and score are both important. Lastly, in Chapter 6, we summarize and conclude from the results.

Chapter 2

Preliminaries

This chapter states the notations and explains the basic concepts used in this thesis. First, the definitions and notations are presented, followed by a description of the structure of the data. Then, we discuss the linear hard margin boundary and the linear soft margin boundary. The nonlinear boundary is examined after that and, finally, there is a description of feature selection in support vector machines.

2.1 Definitions and notations

$[n]$ is used to denote the set $\{1, \ldots, n\}$. For example, to illustrate that n vectors $\mathbf{x}$ are all nonnegative, Equation 2.1 is used:

$$\mathbf{x_i} \geq 0 \quad \forall i \in [n] \tag{2.1}$$

The norm of vector $\mathbf{w}$ is defined as:

$$\|\mathbf{w}\| = \sqrt{\sum_{i=1}^{n} w_i^2} \tag{2.2}$$

To denote the number of elements in a set S, we use:

$$\#S \tag{2.3}$$

$\mathbf{e}$ is used to denote the vector of all ones.

Each set of data consists out of m points $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^n$ and $y \in \{+1, -1\}$. $\mathbf{x}$ contains the values for the n features a point has and y indicates to which class the point belongs. The point belongs to class 1 if $y = 1$, and the point belongs to class 2 if $y = -1$. See Figure 2.1.

A support vector machine tries to solve a binary classification problem. Using a decision function and a decision rule based on training data the model classifies an unseen datapoint. In Figure 2.1, if a new point lies on the lower-left side of the separation boundary, it is classified as class 1, whereas if it lies on the upper-right side, it is classified as class 2.

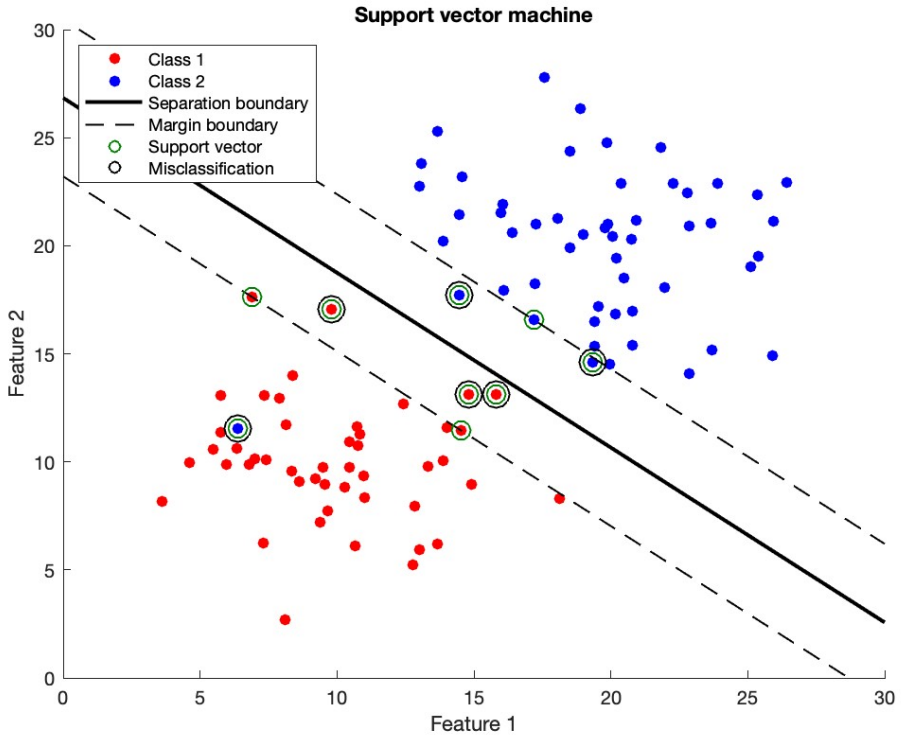

Figure 2.1: A support vector machine solving a binary classification problem. The support vectors and misclassifications are shown. Source: Wim van Duuren.

The decision function is a function d with parameters $\mathbf{x}, \mathbf{w}, b$. With $\mathbf{x}$ being the values for the features of the points, $\mathbf{w}$ the weights for the features and b the bias. The function estimates a value for $\mathbf{y}$. The separation boundary is:

$$d(\mathbf{x}, \mathbf{w}, b) = 0 \tag{2.4}$$

and the decision rule for any point p:

$$p \text{ belongs to class 1 if } d(\mathbf{x_p}, \mathbf{w}, b) > 0 \tag{2.5}$$

$$p \text{ belongs to class 2 if } d(\mathbf{x_p}, \mathbf{w}, b) < 0 \tag{2.6}$$

Since $d(\mathbf{x}, \mathbf{w}, b)$ and $d(\mathbf{x}, k\mathbf{w}, kb)$, with k being a positive scalar, describe the same boundary, a decision function is said to be normal w.r.t. data $\mathbf{x} \in X$ if:

$$\min_{x_i \in X} |\mathbf{w^T x_i} + b| = 1 \tag{2.7}$$

Margin M is the distance between the boundaries parallel to the separating boundary going through the closest members of two classes. See Figure 2.1. Given that point 1 belongs to class 1 and point 2 belongs to class 2, these boundaries are:

$$\begin{aligned} &\mathbf{w^T x_1} - b = 1 \\ &\mathbf{w^T x_2} - b = -1 \\ &\text{unit normal vector: } \frac{\mathbf{w^T}}{\|\mathbf{w}\|} \end{aligned} \tag{2.8}$$

The distance between the boundaries is the scalar projection of the distance between $\mathbf{x_1}$ and $\mathbf{x_2}$ on the unit normal vector.

$$\begin{aligned} M &= \frac{\mathbf{w^T}}{\|\mathbf{w}\|}(\mathbf{x_1} - \mathbf{x_2}) \\ &= \frac{\mathbf{w^T}(\mathbf{x_1} - \mathbf{x_2})}{\|\mathbf{w}\|} \\ &= \frac{\mathbf{w^T x_1} - \mathbf{w^T x_2}}{\|\mathbf{w}\|} \\ &= \frac{\frac{1+b}{\mathbf{w^T}}\mathbf{w^T} - \frac{-1+b}{\mathbf{w^T}}\mathbf{w^T}}{\|\mathbf{w}\|} \\ &= \frac{(1+b) - (-1+b)}{\|\mathbf{w}\|} \\ &= \frac{2}{\|\mathbf{w}\|} \end{aligned} \tag{2.9}$$

Misclassifications in the data are points with less than one distance from the boundary or points on the wrong side of the boundary. This corresponds to:

$$y_i(\mathbf{w^T x_i} + b) < 1 \tag{2.10}$$

The data overlaps when there is no separating boundary, such that there are no misclassified points. The data is linear separable if a linear separating boundary exists, so there are no misclassified points.

The generalization error of a model is a measure of how well the model performs on unseen data. Since a model needs to classify new points correctly, a model with a low generalization error is preferable.

2.2 Linear Hard Margin Boundary

The goal is to find a linear boundary in $\mathbb{R}^n$ that separates the two classes. In this section, we assume that the data does not overlap and is, consequently, linear separable. The decision function is:

$$d(\mathbf{x}, \mathbf{w}, b) = \mathbf{w^T x} + b \tag{2.11}$$

with $\mathbf{w} \in \mathbb{R}^n$ and b being a scalar.

The values of the parameters $\mathbf{w}$ and b are currently unknown and need to be found. The best decision boundary has a maximum margin due to the assumption that data is drawn from an unknown distribution, and a boundary far away from the data is more likely to classify new data drawn from the same distribution correctly (Scholkopf and Smola, 2002). Maxiziming the margin gives good generalization performance (Bousquet et al., 2003). Instead of maximizing the margin M, $\frac{1}{2}\|\mathbf{w}\|^2$ is minimized. This is equivalent since:

$$\max M \implies \min \frac{1}{M} = \min \frac{1}{2}\|\mathbf{w}\| \implies \min \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.12}$$

The separating boundary separates the two classes. To achieve this, we look at the decision rule. Point i belongs to class 1 implies that $y_i = 1$ and $d(\mathbf{x_i}, \mathbf{w}, b) > 0$. The decision rule is rewritten:

$$\begin{aligned} d(\mathbf{x_i}, \mathbf{w}, b) &> 0 \\ y_i d(\mathbf{x_i}, \mathbf{w}, b) &> 0 \\ y_i(\mathbf{w^T x_i} + b) &> 0 \end{aligned} \tag{2.13}$$

Similar for class 2, point i belongs to class 2 implies that $y_i = -1$ and $d(\mathbf{x_i}, \mathbf{w}, b) < 0$. The decision rule is reformulated:

$$\begin{aligned} d(\mathbf{x_i}, \mathbf{w}, b) &< 0 \\ y_i d(\mathbf{x_i}, \mathbf{w}, b) &> 0 \\ y_i(\mathbf{w^T x_i} + b) &> 0 \end{aligned} \tag{2.14}$$

Equation 2.13 and Equation 2.14 show that for both classes $y_i(\mathbf{w^T x_i} + b) > 0$ holds. Thus: $y_i(\mathbf{w^T x_i} + b) = |y_i(\mathbf{w^T x_i} + b)|$. Searching for a normal decision function where $\min_{x_i \in X} |\mathbf{w^T x_i} + b| = 1$, we get: $|\mathbf{w^T x_i} + b| \geq 1$. Since $y_i \in \{1, -1\}$, then: $|y_i| = 1$. Using the Cauchy–Schwarz inequality, we obtain:

$$y_i(\mathbf{w^T x_i} + b) = |y_i(\mathbf{w^T x_i} + b)| \geq |y_i||\mathbf{w^T x_i} + b| = |\mathbf{w^T x_i} + b| \geq 1 \tag{2.15}$$

Therefore, there are restrictions:

$$y_i(\mathbf{w^T x_i} + b) \geq 1 \quad \forall i \in [m] \tag{2.16}$$

with m points.

The optimization problem becomes:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w^T x_i} + b) \geq 1 \quad \forall i \in [m] \\ & \mathbf{w} \in \mathbb{R}^n \\ & b \in \mathbb{R} \end{aligned} \tag{2.17}$$

with m points having n features.

The problem stated in Equation 2.17 is impractical to solve when n and m are large, and no information about the support vectors is obtained (Boser et al., 1992). Hence, the ideas of Boser et al. (1992) and notation from Kecman (2004) are followed to solve these problems. Starting with using the Lagrangian:

$$\begin{aligned} L_p(\mathbf{w}, b, \boldsymbol{\alpha}) \quad &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=i}^{m}(\alpha_i(y_i(\mathbf{w^T x_i} + b) - 1)) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \forall i \in [m] \end{aligned} \tag{2.18}$$

The α_i are the Lagrangian multipliers and satisfy the Kuhn-Tucker conditions:

$$\alpha_i(y_i(\mathbf{w^T x_i} + b) - 1) = 0 \quad \forall i \in [m] \tag{2.19}$$

The saddle point of the Equation 2.18 solves the problem in Equation 2.17. The saddle point is at the minimum of $L_p(\mathbf{w}, b, \boldsymbol{\alpha})$ w.r.t. $\mathbf{w}$ and b (Kecman, 2004), and at the maximum of $L_p(\mathbf{w}, b, \boldsymbol{\alpha})$ w.r.t. $\boldsymbol{\alpha}$. Thus, the saddle point is subject to:

$$\begin{aligned} \frac{\partial L_p}{\partial \mathbf{w}^*} &= \mathbf{w}^* - \sum_{i=1}^{m}(\alpha_i y_i \mathbf{x_i}) = 0 \\ \mathbf{w}^* &= \sum_{i=1}^{m}(\alpha_i y_i \mathbf{x_i}) \\ \frac{\partial L_p}{\partial b^*} &= \sum_{i=1}^{m}(\alpha_i y_i) = 0 \end{aligned} \tag{2.20}$$

From Equation 2.19 and Equation 2.20 follows that the separating boundary $d(\mathbf{x}, \mathbf{w}^*, b^*) = 0$ only depends on points satisfying:

$$y_i(\mathbf{w^{*T}} x_i + b^*) = 1 \tag{2.21}$$

These points are called the support vectors. A subset V with all the support vectors is created:

$$\begin{aligned} \alpha_i &> 0 \quad \forall i \in V \\ y_i(\mathbf{w^{*T}} x_i + b^*) &= 1 \quad \forall i \in V \\ \alpha_i &= 0 \quad \forall i \in V' \\ y_i(\mathbf{w^{*T}} x_i + b^*) &> 1 \quad \forall i \in V' \end{aligned} \tag{2.22}$$

Before solving Equation 2.23, whether a point is a support vector is unknown.

The results from Equation 2.20 are substituted in Equation 2.18 and transformed into the dual problem:

$$\begin{aligned} L_d(\boldsymbol{\alpha}) \quad &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} (y_i y_j \alpha_i \alpha_j \mathbf{x_i^T x_j}) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \forall i \in [m] \\ & \sum_{i=1}^{m}(\alpha_i y_i) = 0 \quad \forall i \in [m] \end{aligned} \tag{2.23}$$

The solution to Equation 2.17 is found when maximizing Equation 2.23 which can be written in matrix notation:

$$\begin{aligned} \max \quad & L_d(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} + \mathbf{e^T}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{y^T}\boldsymbol{\alpha} = 0 \\ & \alpha_i \geq 0 \qquad \forall i \in [m] \end{aligned} \tag{2.24}$$

where $\boldsymbol{\alpha}$ is the vector $[\alpha_1 \dots \alpha_m]^T$, $\mathbf{H}$ is here the matrix with $H_{ij} = y_i y_j \mathbf{x_i^T x_j}$ and $\mathbf{e}$ is the unit vector with size m. When we solve Equation 2.24, with

solution $\boldsymbol{\alpha}^*$, Equation 2.20 is used to obtain:

$$\mathbf{w}^* = \sum_{i=1}^{m}(\alpha_i^* y_i \mathbf{x_i}) \tag{2.25}$$

We use Equation 2.21 and Equation 2.25 to get:

$$\begin{aligned} b^* &= \frac{1}{\#V}\sum_{i=1}^{m}(\frac{1}{y_i} - \mathbf{w}^{*\mathbf{T}}\mathbf{x_i}) \\ &= \frac{1}{\#V}\sum_{i=1}^{m}(y_i - \mathbf{w}^{*\mathbf{T}}\mathbf{x_i}) \\ &= \frac{1}{\#V}\sum_{i=1}^{m}(y_i - (\sum_{\forall i \in V}(\alpha_i^* y_i \mathbf{x_i}))^T \mathbf{x_i}) \\ &= \frac{1}{\#V}\sum_{i=1}^{m}(y_i - \alpha_i^* y_i \mathbf{x_i}^T \mathbf{x_i}) \end{aligned} \tag{2.26}$$

Since $|y_i| = 1$ implies $\frac{1}{y_i} = y_i$. Next, Equation 2.11, Equation 2.25 and Equation 2.26 are used to derive the separating boundary:

$$\begin{aligned} d(\mathbf{x}) &= \mathbf{w}^{*\mathbf{T}}\mathbf{x} + b^* \\ &= (\sum_{i=1}^{m}(\alpha_i^* y_i \mathbf{x_i}))^T \mathbf{x} + \frac{1}{\#V}\sum_{i=1}^{m}(y_i - \alpha_i^* y_i \mathbf{x_i}^T \mathbf{x_i}) \\ &= \sum_{i=1}^{m}(\alpha_i^* y_i \mathbf{x_i}^T \mathbf{x}) + \frac{1}{\#V}\sum_{i=1}^{m}(y_i - \alpha_i^* y_i \mathbf{x_i}^T \mathbf{x_i}) \end{aligned} \tag{2.27}$$

The separating boundary is only based on the support vectors since $\alpha_i^* = 0$ for points that are not a support vector. Support vectors consist generally of a small subset of the training data (Kecman, 2004). This approach reduces the problem's dimensionality, making the calculations more efficient. Another result found by Boser et al. (1992) shows that this approach reduces the model's generalization error. The estimate of the upper bound of the generalization error is obtained by leaving out a different point m times from the training data and measuring how often the left-out point is classified correctly. This method is called leave-one-out cross-validation (LOOCV) (Wong, 2015). If a point is not a support vector, the decision boundary is unchanged, and the point is classified correctly. If the point is a support vector, it can be linearly dependent on the other support vectors and is, therefore, still classified correctly. If the point is a support vector and linearly independent from the other support vectors, the point can be misclassified when left out from the training data. Hence, the estimate of the upper bound of the generalization error is:

$$\frac{\#LinIndep(V)}{m} \leq \frac{\min(n, m)}{m} \tag{2.28}$$

2.3 Linear Soft Margin Boundary

In this section, the assumption that the data has no overlap is dropped. As a result, the constraint as Equation 2.16 in Equation 2.17 can no longer be

satisfied for all points. Thus, misclassifications must be allowed in the model. This is achieved by adding slack variables. The constraint becomes:

$$\begin{aligned} y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) &\geq 1 - \xi_i \quad \forall i \in [m] \\ \xi_i &\geq 0 \quad \forall i \in [m] \end{aligned} \tag{2.29}$$

with ξ_i being the distance between the expected location of point i, assuming that the model is correct, and the actual location of the point given by the data. See Figure 2.1 for a graphical representation of the misclassifications.

The optimization problem becomes:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i \qquad \forall i \in [m] \\ & \xi_i \geq 0 \qquad \forall i \in [m] \\ & \mathbf{w} \in \mathbb{R}^n \\ & b \in \mathbb{R} \end{aligned} \tag{2.30}$$

with m points having n features.

The goal is to penalize the misclassifications. To accomplish this, a cost function to minimize is created. Examples of such a cost function $C(\boldsymbol{\xi})$ are:

$$\begin{aligned} \text{Number of misclassified points} \quad & C(\boldsymbol{\xi}) = \sum_{i=1}^{m} \mathbb{1}_{\xi_i > 0} \\ \text{Sum of distances of the misclassifications} \quad & C(\boldsymbol{\xi}) = \sum_{i=1}^{m} \xi_i \\ \text{Sum of squared distances of the misclassifications} \quad & C(\boldsymbol{\xi}) = \sum_{i=1}^{m} \xi_i^2 \end{aligned} \tag{2.31}$$

The next step is to add the cost function to the objective function of the problem. Multiplying the cost function by a nonnegative scalar c adjusts the penalty assigned to misclassifications. By changing the value of c, we can control how heavily misclassifications are penalized in the objective function. The optimization problem becomes:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + cC(\boldsymbol{\xi}) \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i \qquad \forall i \in [m] \\ & \xi_i \geq 0 \qquad \forall i \in [m] \\ & \mathbf{w} \in \mathbb{R}^n \\ & b \in \mathbb{R} \\ & \boldsymbol{\xi} \in \mathbb{R}^m \end{aligned} \tag{2.32}$$

with m points having n features and c being a nonnegative scalar. An interesting observation is that overlap is again impossible when $C(\boldsymbol{\xi}) = \infty$. Therefore,

the problem may be feasible only for $C(\boldsymbol{\xi}) < \infty$ (Kecman, 2004).

As in Equation 2.17, the problem arises that Equation 2.32 is impractical to solve when n and m are large and no information is gained about the support vectors. Cortes and Vapnik (1995) introduced a general cost function:

$$C(\boldsymbol{\xi}) = \sum_{i=1}^{m} \xi_i^k \tag{2.33}$$

They showed in the case of $k = 1$, which corresponds with the sum of distances in Equation 2.31 and the soft margin SVM corresponding to this cost function is named the L_1 SVM, Equation 2.32 can be solved by transforming the problem into the dual problem. Their ideas and the notation of Kecman (2004) is followed to show this. Starting with the Lagrangian primal:

$$\begin{aligned} &L_p(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \\ &\frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m} \xi_i - \quad \sum_{i=i}^{m}(\alpha_i(y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) - 1 + \xi_i)) - \sum_{i=1}^{m}(\beta_i\xi_i) \\ &\text{s.t.} \qquad\qquad\qquad\qquad\qquad\qquad \alpha_i \geq 0 \quad \forall i \in [m] \\ &\qquad\qquad\qquad\qquad\qquad\qquad \beta_i \geq 0 \quad \forall i \in [m] \end{aligned} \tag{2.34}$$

The α_i and β_i are the Lagrangian multipliers and satisfy the Kuhn-Tucker conditions:

$$\begin{aligned} \alpha_i(y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) - 1 + \xi_i) = 0 \quad \forall i \in [m] \\ \beta_i\xi_i = (c - \alpha_i)\xi_i = 0 \quad \forall i \in [m] \end{aligned} \tag{2.35}$$

Our purpose is to determine the optimal saddle point, as the Lagrangian must be minimized w.r.t. $\mathbf{w}$, b, and $\boldsymbol{\xi}$, while being maximized w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Thus, the saddle point is subject to:

$$\begin{aligned} \frac{\partial L_p}{\partial \mathbf{w}^*} &= \mathbf{w}^* - \sum_{i=1}^{m}(\alpha_i y_i \mathbf{x_i}) = 0 \\ \mathbf{w}^* &= \sum_{i=1}^{m}(\alpha_i y_i \mathbf{x_i}) \\ \frac{\partial L_p}{\partial b^*} &= \sum_{i=1}^{m}(\alpha_i y_i) = 0 \end{aligned} \tag{2.36}$$

and

$$\frac{\partial L_p}{\partial \xi_i^*} = c - \alpha_i - \beta_i = 0 \tag{2.37}$$

Using Equation 2.37, Equation 2.34 is rewritten into:

$$\begin{aligned} &L_p(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \\ &\frac{1}{2}\|\mathbf{w}^*\|^2 + c\sum_{i=1}^{m} \xi_i^* - \sum_{i=i}^{m}(\alpha_i(y_i(\mathbf{w}^{*\mathbf{T}}\mathbf{x_i} + b^*) - 1 + \xi_i^*)) - \sum_{i=1}^{m}(\beta_i\xi_i^*) = \\ &\frac{1}{2}\|\mathbf{w}^*\|^2 - \sum_{i=i}^{m}(\alpha_i(y_i(\mathbf{w}^{*\mathbf{T}}\mathbf{x_i} + b^*) - 1)) - \sum_{i=1}^{m}((c - \alpha_i - \beta_i)\xi_i^*) = \\ &\frac{1}{2}\|\mathbf{w}^*\|^2 - \sum_{i=i}^{m}(\alpha_i(y_i(\mathbf{w}^{*\mathbf{T}}\mathbf{x_i} + b^*) - 1)) \end{aligned} \tag{2.38}$$

The Lagrangian does not depend on $\boldsymbol{\beta}$. Using Equation 2.36 and Equation 2.37, Equation 2.38 is transformed into dual form:

$$\begin{aligned} L_d(\boldsymbol{\alpha}) \quad &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} (y_i y_j \alpha_i \alpha_j \mathbf{x_i^T x_j}) \\ \text{s.t.} \qquad & \sum_{i=1}^{m} (\alpha_i y_i) = 0 \quad \forall i \in [m] \\ & c \geq \alpha_i \geq 0 \quad \forall i \in [m] \end{aligned} \tag{2.39}$$

This is similar to Equation 2.23, except that the α_i's are now bounded from above by c. Points with $\alpha_i > 0$ are still support vectors. Points with $c > \alpha_i > 0$ lie on the margin, and points with $\alpha_i = c$ lie on the wrong side of the margin. Points lying on the wrong side of the margin with $\xi_i \geq 1$ are misclassified. See Figure 2.1.

The solution is found by maximizing Equation 2.39, which can be written in matrix notation:

$$\begin{aligned} \max \quad & L_d(\boldsymbol{\alpha}) = -\frac{1}{2} \boldsymbol{\alpha^T} \mathbf{H} \boldsymbol{\alpha} + \mathbf{e^T} \boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{y^T} \boldsymbol{\alpha} = 0 \\ & c \geq \alpha_i \geq 0 \qquad \forall i \in [m] \end{aligned} \tag{2.40}$$

where $\boldsymbol{\alpha}$ is the vector $[\alpha_1 \dots \alpha_m]^T$, $\mathbf{H}$ is here the matrix with $H_{ij} = y_i y_j \mathbf{x_i^T x_j}$ and $\mathbf{e}$ is the unit vector with size m. The L_1 SVM is further solved similarly as described after Equation 2.24.

From Lin (2001), we take the dual formulation for $k = 2$, called the L_2 SVM:

$$\begin{aligned} \max \quad & L_d(\boldsymbol{\alpha}) = -\frac{1}{2} \boldsymbol{\alpha^T} (\mathbf{H} + \frac{I}{c}) \boldsymbol{\alpha} + \mathbf{e^T} \boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{y^T} \boldsymbol{\alpha} = 0 \\ & \alpha_i \geq 0 \qquad \forall i \in [m] \end{aligned} \tag{2.41}$$

with identity matrix I, $\boldsymbol{\alpha}$ is the vector $[\alpha_1 \dots \alpha_m]^T$, $\mathbf{H}$ is here the matrix with $H_{ij} = y_i y_j \mathbf{x_i^T x_j}$ and $\mathbf{e}$ is the unit vector with size m. The L_2 SVM is further solved in similar fashion as described after Equation 2.24.

Comparing the L_1 SVM and L_2 SVM, Abe (2005) finds that the L_1 SVM creates a separating boundary with fewer support vectors. This is preferable from a computational standpoint.

2.4 Nonlinear Boundary

The previous sections were limited to a linear boundary separating the data. The set of linear boundaries is a small subset of all boundaries. This section

generalizes the separating boundaries by including nonlinear boundaries. We use the kernel trick, a method invented by Boser et al. (1992), to achieve this. The ideas of Boser et al. (1992) and notation by Kecman (2004) are adopted in this section. The kernel trick can be applied to both the hard and soft margin boundary. The essential requirement is to be able to write the problem in dual form. The method is shown for the L_1 SVM.

First, $\mathbf{x_i} \in \mathbb{R}^n$ is mapped to $\phi(\mathbf{x_i}) \in \mathbb{R}^h$, with $h > n$. Then, the problem is linearized by replacing $\mathbf{x_i}$ with $\phi(\mathbf{x_i})$ and solved in $\mathbb{R}^h$. This leads to a linear boundary in $\mathbb{R}^h$ and a nonlinear boundary in $\mathbb{R}^n$. The issue with this approach is that calculating the product $\phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$ becomes difficult when n becomes large. The kernel trick solves this. Instead of first calculating $\phi(\mathbf{x_i})$ and $\phi(\mathbf{x_j})$, and then calculating the product between the two vectors to obtain a scalar, a kernel function $K(\mathbf{x_i}, \mathbf{x_j})$ is used to directly calculate the scalar without first mapping the vectors. For this, $\phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$ is replaced by the function $K(\mathbf{x_i}, \mathbf{x_j})$ with mapping $\mathbb{R}^n \to \mathbb{R}$. An example of this kernel function is the polynomial function of degree d: $K(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i^T x_j} + 1)^d$. To use the kernel function in the L_1 SVM, $\mathbf{x_i^T x_j}$ is replaced by $K(\mathbf{x_i}, \mathbf{x_j})$ in Equation 2.39 to obtain:

$$\begin{aligned} L_d(\boldsymbol{\alpha}) \quad &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} (y_i y_j \alpha_i \alpha_j K(\mathbf{x_i}, \mathbf{x_j})) \\ \text{s.t.} \qquad & c \geq \alpha_i \geq 0 \quad \forall i \in [m] \\ & \sum_{i=1}^{m} (\alpha_i y_i) = 0 \quad \forall i \in [m] \end{aligned} \tag{2.42}$$

with solution:

$$d(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i^* y_i K(\mathbf{x_i}, \mathbf{x})) + b^* \tag{2.43}$$

b^* is calculated by noticing that $d(\mathbf{x_1}) = 1$ with $\mathbf{x_1} \in$ Class 1 and $d(\mathbf{x_2}) = -1$ with $\mathbf{x_2} \in$ Class 2:

$$\begin{aligned} 0 = d(\mathbf{x_1}) + d(\mathbf{x_2}) &= \sum_{i=1}^{m} (\alpha_i^* y_i K(\mathbf{x_i}, \mathbf{x_1})) + \sum_{i=1}^{m} (\alpha_i^* y_i K(\mathbf{x_i}, \mathbf{x_2})) + 2b^* \\ b^* &= -\frac{1}{2} (\sum_{i=1}^{m} (\alpha_i^* y_i (K(\mathbf{x_i}, \mathbf{x_1}) + K(\mathbf{x_i}, \mathbf{x_2}))) \end{aligned} \tag{2.44}$$

2.5 Feature Selection

Up till now, we covered functions that penalize the misclassifications. Since Mangasarian and Bradley (1998), there have been proposals to penalize the weights of the features in an SVM setting. The idea is that, by selecting a subset of features, the dimensionality of the feature space reduces, and the model generalizes better. The generalization error is, therefore, lower. By penalizing some weights, a penalty function gives different importance to different features. By setting certain weights to zero, the function selects the features corresponding to the non-zero weights to be included only in the separating

boundary. This is done by substituting $\frac{1}{2}\|\mathbf{w}\|^2$ with the more general function $p(\mathbf{w})$ in the objective function.

Below we present three functions, prosed by Mangasarian and Bradley (1998), using notation from Neumann et al. (2005):

The original L_2-norm, without the constant $\frac{1}{2}$ it was first proposed by Hoerl and Kennard (1979) in a non-SVM setting and named the Ridge penalty:

$$p(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.45}$$

The L_1-norm, named LASSO (least absolute shrinkage and selection operator) and first proposed in a non-SVM setting by Tibishirani (1996):

$$p(\mathbf{w}) = \mathbf{e}^{\mathbf{T}}|\mathbf{w}| \tag{2.46}$$

The feature selection concave, which approximates the L_0-norm:

$$p(\mathbf{w}) = \mathbf{e}^{\mathbf{T}}(\mathbf{e} - e^{-\alpha|\mathbf{w}|}) \tag{2.47}$$

with $\alpha \in \mathbb{R}_+$.

According to Neumann et al. (2005) "the L_2 penalty term is responsible for the very good SVM classification results while the L_1 and L_0 penalty terms focus on feature selection" (p. 133). The L_2 penalty does not set any weights to zero, and the L_1 and L_0 penalties do (Becker et al., 2011). They proposed the following combined penalty functions with weight parameter $\mu \in (0, 1)$:

L_2L_1, simultaneously proposed by Zou and Hastie (2005) as Elastic Net:

$$p(\mathbf{w}) = \mu\frac{1}{2}\|\mathbf{w}\|^2 + (1-\mu)\mathbf{e}^{\mathbf{T}}|\mathbf{w}| \tag{2.48}$$

L_2L_0:

$$p(\mathbf{w}) = \mu\frac{1}{2}\|\mathbf{w}\|^2 + (1-\mu)\mathbf{e}^{\mathbf{T}}(\mathbf{e} - e^{-\alpha|\mathbf{w}|}) \tag{2.49}$$

The authors reported improved performance compared to the penalty functions mentioned in Equation 2.45, Equation 2.46 and Equation 2.47.

Another relevant function is the smoothly clipped absolute deviation (SCAD) penalty function proposed in Fan (1997) in the context of wavelet applications in statistics. The concept was named SCAD and worked out further in Fan and Li (2001). SCAD was adapted to the SVM in Zhang et al. (2006). The SCAD penalty function is:

$$p_\lambda(|w|) = \begin{cases} \lambda|w| & \text{if } |w| \leq \lambda \\ -\frac{|w|^2 - 2\alpha\lambda|w| + \lambda^2}{2(\alpha-1)} & \text{if } \lambda < |w| \leq \alpha\lambda \\ \frac{(\alpha+1)\lambda^2}{2} & \text{if } |w| > \alpha\lambda \end{cases} \tag{2.50}$$

with $\alpha > 2$ and $\lambda > 0$ as tuning parameters. Applied to the SVM we have:

$$p(\mathbf{w}) = \sum_{j=1}^{n} p_\lambda(|w_j|) \tag{2.51}$$

Zhang et al. (2006) showed that the SCAD penalty function performs better in sparse settings than the L_1-norm and L_2-norm compared at average error rate, number of variables selected and frequency of selecting correct variables.

According to Becker et al. (2011) "the SCAD penalty might be too strict in selecting features for non-sparse data" (p. 4). They proposed a combination of SCAD and L_2. Dubbed the Elastic SCAD, the penalty function is:

$$p(\mathbf{w}) = \mu \sum_{j=1}^{n} p_\lambda(|w_j|) + (1-\mu)\frac{1}{2}\|\mathbf{w}\|^2 \tag{2.52}$$

with weight parameter $\mu \in (0, 1)$. The authors found similar results as Zhang et al. (2006). In addition, they found that in non-sparse settings, Elastic SCAD performed better compared at the average error rate, number of variables selected and frequency of selecting correct variables. The Elastic SCAD selected fewer features than the penalty function in Equation 2.48. Algorithms to rapidly find an optimal solution using SCAD and Elastic SCAD are proposed by Becker et al. (2011) and more recently by Al-Thanoon et al. (2018). However, this thesis does not use these methods.

This chapter contains the basic concepts of this thesis. The linear soft margin support vector machine is the basis for the models in Chapter 4. If a model can be written in dual form, it is possible to obtain information about the support vectors and the kernel trick can be applied. The penalty functions selecting features are used as cost functions dealing with misclassifications in a few new models. In the next chapter, the simulation of the datasets is explained.

Chapter 3

Data

This chapter proposes two methods of simulating the data. The first gives more control in measurements, such as overlap and outliers, while the second gives more freedom in how data is drawn.

3.1 Uniformly distributed

Parameter	Description
m	Data points
n	Number of features
scale	Length of the interval data is drawn from
overlap	Share of the data that overlaps
noise	Share of the training data that is noise
outlier	Ratio between scale of the noise and *scale*

Table 3.1: Parameter descriptions for D1

The values for a single feature are drawn from a uniform distribution in the interval (0, *scale*) for one class and for the other class in the interval ($-scale$, 0). We have uniform distributions:

$$\begin{aligned} &\mathcal{U}(0, scale) \\ &\mathcal{U}(-scale, 0) \end{aligned} \tag{3.1}$$

If the share of the data that overlaps needs to be *overlap*, the intervals are adjusted to:

$$\begin{aligned} &\mathcal{U}(-\frac{1}{2}scale\sqrt[n]{overlap}, scale - \frac{1}{2}scale\sqrt[n]{overlap}) \\ &\mathcal{U}(-scale + \frac{1}{2}scale\sqrt[n]{overlap}, \frac{1}{2}scale\sqrt[n]{overlap}) \end{aligned} \tag{3.2}$$

Then a single feature overlaps with $\sqrt[n]{overlap}$ and the total data with $(\sqrt[n]{overlap})^n = overlap$. For convenience $\frac{1}{2}scale\sqrt[n]{overlap}$ is replaced by l:

$$\begin{aligned} &\mathcal{U}(-l, scale - l) \\ &\mathcal{U}(-scale + l, l) \end{aligned} \tag{3.3}$$

First, a random set F of size $\leq n$ is generated containing random indexes of the features. Then, we create sets $Class1$ and $Class2$ with each half of the indexes of the points. Both sets have size m.

$$\begin{aligned} y_i &= 1 && \forall i \in Class1 \\ y_i &= -1 && \forall i \in Class2 \\ x_{ij} &\sim \mathcal{U}(-scale + l, l) && \forall i \in Class1, \forall j \in F \\ x_{ij} &\sim \mathcal{U}(-l, scale - l) && \forall i \in Class1, \forall j \in F' \\ x_{ij} &\sim \mathcal{U}(-scale + l, l) && \forall i \in Class2, \forall j \in F' \\ x_{ij} &\sim \mathcal{U}(-l, scale - l) && \forall i \in Class2, \forall j \in F \end{aligned} \tag{3.4}$$

with index i for point i and index j for feature j.

All points are divided equally into two sets of size m, the $Training$ set and the $Test$ set. Random set E contains $noise\%$ random indexes for the training data points. These entries are replaced by:

$$x_{ij} \sim \mathcal{U}((-scale + l)outlier, 2(scale - l)outlier) \quad \forall i \in E, \forall j \in [n] \tag{3.5}$$

Then, two random features are rotated around the origin. The rotation matrix is the identity matrix of size $n \times n$ where random entries are altered as:

$$\begin{bmatrix} I_{ii} & I_{ij} \\ I_{ji} & I_{jj} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.6}$$

with $i < j$. This procedure is repeated $n - 1$ times. The same rotation matrix is used for the training and test sets for every rotation.

After the rotation, all values for x are shifted with $shift \sim \mathcal{U}(-scale, scale)$:

$$x_{ij} \rightarrow x_{ij} + shift \tag{3.7}$$

$Training$ and $Test$ are then complete.

3.2 Multiple distributions

As a start, sets $Class1$ and $Class2$ are created with each half of the indexes of the points. Both sets have size m.

$$\begin{aligned} y_i &= 1 && \forall i \in Class1 \\ y_i &= -1 && \forall i \in Class2 \\ x_{ij} &\sim pdSet(j) && \forall i \in Class1, \forall j \in [n] \\ x_{ij} &\sim pdSet(n + j) && \forall i \in Class2, \forall j \in [n] \end{aligned} \tag{3.8}$$

All points are divided equally into two sets of size $m \times n$, the $Training$ set and the $Test$ set. $Test$ is now completed. We generate random sets $E1$ containing

Parameter	Description
m	Data points
n	Number of features
$pdSet$	Set of probability distribution of size $2n$ for the correct data
$noisePdSet$	Set of probability distribution of size $2n$ for the noise data
$noise1$	Share of the training data of class 1 that is noise
$noise2$	Share of the training data of class 2 that is noise

Table 3.2: Parameter descriptions for D2

$noise1\%$ random indexes for the training data points from class 1 and $E2$ containing $noise2\%$ random indexes for the training data points from class 2. The entries are substituted by:

$$\begin{aligned} x_{ij} &\sim noisePdSet(j) && \forall i \in E1, \forall j \in [n] \\ x_{ij} &\sim noisePdSet(n+j) && \forall i \in E2, \forall j \in [n] \end{aligned} \tag{3.9}$$

$Training$ is then complete.

For every combination of parameter values tested, for both types of datasets, thirty datasets are generated. This way, the mean score, standard deviation and mean runtime can be calculated. The objective is to reach a conclusion by examining the performance of the models on the general problem rather than on a specific dataset. To achieve this, we choose a parameter and generate datasets with at least nine different values for it, while keeping all other parameters constant. This is repeated for all numerical parameters resulting in $79 \times 30 = 2370$ datasets.

This chapter explained how the datasets, on which the models are tested, are generated. The next chapter presents the models.

Chapter 4

Models

We start this chapter by describing the models used to compare our new models. These models are the benchmark models, and they use the cost functions from Equation 2.31. A method based on bootstrapping, a model using a capped L_1 cost function and models using cost functions from Section 2.5 are then proposed. For each model, we question if that model can be written in dual form. It is preferable to write the models in dual form so that the support vectors can be retrieved and the kernel trick can be applied. To obtain results for the models, we use the Gurobi solver in Matlab. The Gurobi interface has some restrictions; some models need to be rewritten for them to work with this interface. The interface does not allow for indicator functions, third- and higher-order functions and exponential functions with a constant in the exponent. It also does not allow for exponential functions in the objective function. Two implementations of the L_0 SVM are presented and differ in whether a binary variable is used. L_0^a and L_0^b are used to indicate the different models. Plots for the cost functions can be found in Appendix A.

4.1 Benchmark Models

4.1.1 L_0^a SVM

The L_0^a SVM uses the number of misclassified points from Equation 2.31 as a cost function. For a plot of this function, see Figure A.1. The model is:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m} \mathbb{1}_{\xi_i>0} & & \\ \text{s.t.} \quad & y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i & & \forall i \in [m] \\ & \xi_i \geq 0 & & \forall i \in [m] \\ & \mathbf{w} \in \mathbb{R}^n & & \\ & b \in \mathbb{R} & & \\ & \boldsymbol{\xi} \in \mathbb{R}^m & & \end{aligned} \tag{4.1}$$

Since the Gurobi interface cannot use an indicator function, the model must be rewritten using a binary variable and a large constant M. Using $\mathbf{x} \in \mathbb{R}^{n\times m}$,

$\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$ and large $M \in \mathbb{R}^+$, the L_0^a SVM then becomes:

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m} z_i \\
\text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i \qquad \forall i \in [m] \\
& \xi_i \geq 0 \qquad \forall i \in [m] \\
& -\xi_i + z_i M \geq 0 \qquad \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n \\
& b \in \mathbb{R} \\
& \boldsymbol{\xi} \in \mathbb{R}^m \\
& \mathbf{z} \in \{0,1\}^m
\end{aligned} \tag{4.2}$$

4.1.2 L_1 SVM

Using $\mathbf{x} \in \mathbb{R}^{n\times m}$, $\mathbf{y} \in \mathbb{R}^m$ and $c \in \mathbb{R}^+$, the L_1 SVM uses the sum of distances of the misclassifications from Equation 2.31 as a cost function:

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m} \xi_i \\
\text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i \qquad \forall i \in [m] \\
& \xi_i \geq 0 \qquad \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n \\
& b \in \mathbb{R} \\
& \boldsymbol{\xi} \in \mathbb{R}^m
\end{aligned} \tag{4.3}$$

We can write this model in the dual form as in Equation 2.40. See Figure A.2 for a visualization of the cost function.

4.1.3 L_2 SVM

The L_2 SVM uses the sum of squared distances of the misclassifications from Equation 2.31 as a cost function, using $\mathbf{x} \in \mathbb{R}^{n\times m}$, $\mathbf{y} \in \mathbb{R}^m$ and $c \in \mathbb{R}^+$:

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m} \xi_i^2 \\
\text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i \qquad \forall i \in [m] \\
& \xi_i \geq 0 \qquad \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n \\
& b \in \mathbb{R} \\
& \boldsymbol{\xi} \in \mathbb{R}^m
\end{aligned} \tag{4.4}$$

This model can be written in the dual form as in Equation 2.41. For the plotted cost function, see Figure A.3.

4.2 Proposed Models

4.2.1 Method based on bootstrapping

One of the shortcomings of all three benchmark models is that their performance is sensitive to noisy data (Li et al., 2013). All three models penalize misclassifications, even when those misclassifications are noise in the data. In that case, these points are wrongly penalized. Our proposal is using a method based on bootstrapping: a subset of points of the training data is taken, possibly containing noise. With that subset, we use the L_1 SVM to calculate the separating boundary. Then, the points that are not in the subset are reclassified, possibly reducing the data noise. This is repeated for some iterations. The separating boundary found in the last iteration solves the original problem. Using $\mathbf{x} \in \mathbb{R}^{n \times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$, $t \in \mathbb{N}$, and $\alpha \in (0, 1)$, we incorporated this in the method based on bootstrapping:

Repeat t times:

Step 1:
Generate a random subset S with $\alpha\%$ of the indexes of $Training$. Make sure that S contains the same number of indexes corresponding to class 1 points as to class 2 points.

Step 2:
Use the L_1 SVM with S as data.

Step 3:

$$\forall i \in S' : y_i = \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x}_i + b \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{4.5}$$

The method based on bootstrapping can still be written in dual form as in Equation 2.40 since it uses the L_1 SVM.

4.2.2 Capped L_1 SVM

Since the L_1 SVM and the L_2 SVM use the distance in the cost function, they struggle dealing with outliers. A handful of extreme outliers can skew the results, making all the other points irrelevant. The L_2 SVM is more affected by this problem than the L_1 SVM since outliers become relatively more penalized than other points when using the squared distance. To lessen the effect of outliers on the result, a maximum is put on the value a single misclassification

can have in the cost function. See Figure A.4. The capped L_1 SVM is:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}\min(\xi_i, d) & \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i & \forall i \in [m] \\ & \xi_i \geq 0 & \forall i \in [m] \\ & \mathbf{w} \in \mathbb{R}^n & \\ & b \in \mathbb{R} & \end{aligned} \tag{4.6}$$

Gurobi cannot work with $\min(\xi_i, d)$, so the SVM is changed into a binary problem:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(z_i\xi_i + (1 - z_i)d) & \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i & \forall i \in [m] \\ & \xi_i \geq 0 & \forall i \in [m] \\ & \xi_i - d + (1 - z_i)M \geq 0 & \forall i \in [m] \\ & d - \xi_i + z_i M \geq 0 & \forall i \in [m] \\ & \mathbf{w} \in \mathbb{R}^n & \\ & b \in \mathbb{R} & \\ & \boldsymbol{\xi} \in \mathbb{R}^m & \\ & \mathbf{z} \in \{0, 1\}^m & \end{aligned} \tag{4.7}$$

This equation is reformulated into an easier solvable problem, using $\mathbf{x} \in \mathbb{R}^{n\times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$, $d \in \mathbb{R}^+$ and large $M \in \mathbb{R}^+$:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(u_i + (1 - z_i)d) & \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i & \forall i \in [m] \\ & \xi_i \geq 0 & \forall i \in [m] \\ & \xi_i - d + (1 - z_i)M \geq 0 & \forall i \in [m] \\ & d - \xi_i + z_i M \geq 0 & \forall i \in [m] \\ & u_i \leq z_i M & \forall i \in [m] \\ & u_i \leq \xi_i & \forall i \in [m] \\ & u_i \geq \xi_i - (1 - z_i)M & \forall i \in [m] \\ & u_i \geq 0 & \forall i \in [m] \\ & \mathbf{w} \in \mathbb{R}^n & \\ & b \in \mathbb{R} & \\ & \boldsymbol{\xi} \in \mathbb{R}^m & \\ & \mathbf{z} \in \{0, 1\}^m & \\ & \mathbf{u} \in \mathbb{R}^m & \end{aligned} \tag{4.8}$$

We cannot write the capped L_1 SVM into dual form.

4.2.3 L_0^b SVM

The following models use cost functions from the field of feature selection mentioned in Section 2.5. We did not encountere any literature applying these functions on the misclassification instead of on the weight vector. The models can, therefore, be considered novel. For all hold that we cannot write them in dual form. The first model uses the cost function, see Figure A.5 for a visualization, from Equation 2.47 applied to the misclassifications:

$$C(\boldsymbol{\xi}) = \mathbf{e}^{\mathbf{T}}(\mathbf{e} - e^{-\alpha|\boldsymbol{\xi}|}) = \sum_{i=1}^{m}(1 - e^{-\alpha|\xi_i|}) \tag{4.9}$$

The L_0^b SVM then becomes:

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(1 - u_i) & \\
\text{s.t.} \quad & y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i & \forall i \in [m] \\
& \xi_i \geq 0 & \forall i \in [m] \\
& u_i = e^{p_i} & \forall i \in [m] \\
& p_i = -\alpha\xi_i & \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n & \\
& b \in \mathbb{R} & \\
& \boldsymbol{\xi} \in \mathbb{R}^m & \\
& \mathbf{u} \in \mathbb{R}^m & \\
& \mathbf{p} \in \mathbb{R}^m &
\end{aligned} \tag{4.10}$$

A way to obtain results more easily, is to write the equation using $\mathbf{x} \in \mathbb{R}^{n\times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$ and $\alpha \in \mathbb{R}^+$:

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(1 - u_i) & \\
\text{s.t.} \quad & y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i & \forall i \in [m] \\
& \xi_i \geq 0 & \forall i \in [m] \\
& u_i \leq e^{p_i} & \forall i \in [m] \\
& p_i = -\alpha\xi_i & \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n & \\
& b \in \mathbb{R} & \\
& \boldsymbol{\xi} \in \mathbb{R}^m & \\
& \mathbf{u} \in \mathbb{R}^m & \\
& \mathbf{p} \in \mathbb{R}^m &
\end{aligned} \tag{4.11}$$

4.2.4 L_2L_1 SVM

The L_2L_1 SVM uses the cost function from Equation 2.48:

$$C(\boldsymbol{\xi}) = c\sum_{i=1}^{m}(\mu\xi_i^2 + (1 - \mu)\xi_i) \tag{4.12}$$

Inserting this function using $\mathbf{x} \in \mathbb{R}^{n \times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$ and $\mu \in \mathbb{R} \cap (0, 1)$, we have:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(\mu\xi_i^2 + (1-\mu)\xi_i) & & \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i & \forall i \in [m] & \\ & \xi_i \geq 0 & \forall i \in [m] & (4.13) \\ & \mathbf{w} \in \mathbb{R}^n & & \\ & b \in \mathbb{R} & & \\ & \boldsymbol{\xi} \in \mathbb{R}^m & & \end{aligned}$$

4.2.5 $L_2L_0^b$ SVM

Using the cost function from Equation 2.49:

$$C(\boldsymbol{\xi}) = \sum_{i=1}^{m}(\mu\xi_i^2 + (1-\mu)(1 - e^{-\alpha|\xi_i|})) \tag{4.14}$$

Altering Equation 4.11 with Equation 4.14 and using $\mathbf{x} \in \mathbb{R}^{n \times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$ and $\mu \in \mathbb{R} \cap (0, 1)$, we have the $L_2L_0^b$ SVM:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(\mu\xi_i^2 + (1-\mu)(1-u_i)) & & \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathbf{T}\mathbf{x_i} + b) \geq 1 - \xi_i & \forall i \in [m] & \\ & \xi_i \geq 0 & \forall i \in [m] & \\ & u_i \leq e^{p_i} & \forall i \in [m] & \\ & p_i = -\alpha\xi_i & \forall i \in [m] & (4.15) \\ & \mathbf{w} \in \mathbb{R}^n & & \\ & b \in \mathbb{R} & & \\ & \boldsymbol{\xi} \in \mathbb{R}^m & & \\ & \mathbf{u} \in \mathbb{R}^m & & \\ & \mathbf{p} \in \mathbb{R}^m & & \end{aligned}$$

4.2.6 $L_1L_0^b$ SVM

Inspired by the L_2L_1 SVM and the $L_2L_0^b$ SVM, we create a model using a combination of the L_1 and the L_0^b cost function, named the $L_1L_0^b$ SVM. Adapting Equation 4.11 to this cost functions and using $\mathbf{x} \in \mathbb{R}^{n \times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$ and

$\mu \in \mathbb{R} \cap (0,1)$, we have:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(\xi_i + \mu(1-u_i)) & & \\
\text{s.t.} \quad & y_i(\mathbf{w^T x_i} + b) \geq 1 - \xi_i & \forall i \in [m] & \\
& \xi_i \geq 0 & \forall i \in [m] & \\
& u_i \leq e^{p_i} & \forall i \in [m] & \\
& p_i = -\alpha\xi_i & \forall i \in [m] & \qquad (4.16) \\
& \mathbf{w} \in \mathbb{R}^n & & \\
& b \in \mathbb{R} & & \\
& \boldsymbol{\xi} \in \mathbb{R}^m & & \\
& \mathbf{u} \in \mathbb{R}^m & & \\
& \mathbf{p} \in \mathbb{R}^m & &
\end{aligned}
$$

4.2.7 SCAD SVM

The SCAD SVM uses Equation 2.51 as:

$$C(\boldsymbol{\xi}) = \sum_{i=1}^{m} p_\lambda(|\xi_i|) = \sum_{i=1}^{m}(\mathbb{1}_{|\xi_i|\leq\lambda}\lambda|\xi_i| - \mathbb{1}_{\lambda<|\xi_i|\leq\alpha\lambda}\frac{|\xi_i|^2 - 2\alpha\lambda|\xi_i| + \lambda^2}{2(\alpha-1)} + \mathbb{1}_{|\xi_i|>\alpha\lambda}\frac{(\alpha+1)\lambda^2}{2}) \tag{4.17}$$

See a graphical representation in Figure A.6. Gurobi does not allow for indicator functions, therefore:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(z_i\lambda\xi_i - q_i(1-z_i)\frac{\xi_i^2 - 2\alpha\lambda\xi_i + \lambda^2}{2(\alpha-1)} + (1-q_i)\frac{(\alpha+1)\lambda^2}{2}) & & \\
\text{s.t.} \quad & y_i(\mathbf{w^T x_i} + b) \geq 1 - \xi_i & \forall i \in [m] & \\
& \xi_i \geq 0 & \forall i \in [m] & \\
& \lambda - \xi_i + (1-z_i)M \geq 0 & \forall i \in [m] & \\
& \xi_i - \lambda + z_i M \geq 0 & \forall i \in [m] & \\
& \alpha\lambda - \xi_i + (1-q_i)M \geq 0 & \forall i \in [m] & \\
& \xi_i - \alpha\lambda + q_i M \geq 0 & \forall i \in [m] & \\
& \mathbf{w} \in \mathbb{R}^n & & \\
& b \in \mathbb{R} & & \\
& \boldsymbol{\xi} \in \mathbb{R}^m & & \\
& \mathbf{z} \in \{0,1\}^m & & \\
& \mathbf{q} \in \{0,1\}^m & & \qquad (4.18)
\end{aligned}
$$

Then, the model has multiple third- or higher-order functions, such as $q_i z_i \xi_i^2$. We write this into the SCAD SVM:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(z_i\lambda\xi_i - (q_i - p_i)\frac{u_i - 2\alpha\lambda\xi_i + \lambda^2}{2(\alpha-1)} + (1-q_i)\frac{(\alpha+1)\lambda^2}{2}) & \\
\text{s.t.} \quad & y_i(\mathbf{w^T x_i} + b) \geq 1 - \xi_i & \forall i \in [m] \\
& \xi_i \geq 0 & \forall i \in [m] \\
& \lambda - \xi_i + (1 - z_i)M \geq 0 & \forall i \in [m] \\
& \xi_i - \lambda + z_i M \geq 0 & \forall i \in [m] \\
& \alpha\lambda - \xi_i + (1 - q_i)M \geq 0 & \forall i \in [m] \\
& \xi_i - \alpha\lambda + q_i M \geq 0 & \forall i \in [m] \\
& q_i z_i = p_i & \forall i \in [m] \\
& \xi_i^2 = u_i & \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n & \\
& b \in \mathbb{R} & \\
& \boldsymbol{\xi} \in \mathbb{R}^m & \\
& \mathbf{z} \in \{0,1\}^m & \\
& \mathbf{q} \in \{0,1\}^m & \\
& \mathbf{p} \in \mathbb{R}^m & \\
& \mathbf{u} \in \mathbb{R}^m &
\end{aligned}
\tag{4.19}
$$

Once again, it is possible to find a way to obtain results for this model more easily using $\mathbf{x} \in \mathbb{R}^{n\times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$, $\lambda \in \mathbb{R}^+$, $\alpha \in \mathbb{R}^+ \cap \alpha > 2$ and large $M \in \mathbb{R}^+$:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(\lambda j_i - \frac{k_i - p_i u_i - 2\alpha\lambda(l_i - p_i\xi_i) + \lambda^2(q_i - p_i)}{2(\alpha-1)} \\
& + (1-q_i)\frac{(\alpha+1)\lambda^2}{2}) \\
\text{s.t.} \quad & y_i(\mathbf{w^T x_i} + b) \geq 1 - \xi_i && \forall i \in [m] \\
& \xi_i \geq 0 && \forall i \in [m] \\
& \lambda - \xi_i + (1-z_i)M \geq 0 && \forall i \in [m] \\
& \xi_i - \lambda + z_i M \geq 0 && \forall i \in [m] \\
& \alpha\lambda - \xi_i + (1-q_i)M \geq 0 && \forall i \in [m] \\
& \xi_i - \alpha\lambda + q_i M \geq 0 && \forall i \in [m] \\
& p_i \leq q_i && \forall i \in [m] \\
& p_i \leq z_i && \forall i \in [m] \\
& p_i \geq q_i + z_i - 1 && \forall i \in [m] \\
& \xi_i^2 = u_i && \forall i \in [m] \\
& j_i \leq z_i M && \forall i \in [m] \\
& j_i \leq \xi_i && \forall i \in [m] \\
& j_i \geq \xi_i - (1-z_i)M && \forall i \in [m] \\
& j_i \geq 0 && \forall i \in [m] \\
& k_i \leq q_i M && \forall i \in [m] \\
& k_i \leq u_i && \forall i \in [m] \\
& k_i \geq u_i - (1-q_i)M && \forall i \in [m] \\
& k_i \geq 0 && \forall i \in [m] \\
& l_i \leq q_i M && \forall i \in [m] \\
& l_i \leq \xi_i && \forall i \in [m] \\
& l_i \geq \xi_i - (1-q_i)M && \forall i \in [m] \\
& l_i \geq 0 && \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n \\
& b \in \mathbb{R} \\
& \boldsymbol{\xi} \in \mathbb{R}^m \\
& \mathbf{z} \in \{0,1\}^m \\
& \mathbf{q} \in \{0,1\}^m \\
& \mathbf{p} \in \mathbb{R}^m \\
& \mathbf{u} \in \mathbb{R}^m \\
& \mathbf{j} \in \mathbb{R}^m \\
& \mathbf{k} \in \mathbb{R}^m \\
& \mathbf{l} \in \mathbb{R}^m
\end{aligned}
\tag{4.20}
$$

4.2.8 Elastic SCAD SVM

The Elastic SCAD SVM uses the cost function from Equation 2.52 and extends the SCAD SVM from Equation 4.20, using $\mathbf{x} \in \mathbb{R}^{n \times m}$, $\mathbf{y} \in \mathbb{R}^m$, $c \in \mathbb{R}^+$, $\lambda \in \mathbb{R}^+$, $\alpha \in \mathbb{R}^+ \cap \alpha > 2$, $\mu \in \mathbb{R} \cap (0, 1)$ and large $M \in \mathbb{R}^+$:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{m}(\mu(\lambda j_i - \frac{k_i - p_i u_i - 2\alpha\lambda(l_i - p_i\xi_i) + \lambda^2(q_i - p_i)}{2(\alpha - 1)} \\
& + (1 - q_i)\frac{(\alpha + 1)\lambda^2}{2}) + (1 - \mu)\xi_i^2) \\
\text{s.t.} \quad & y_i(\mathbf{w^T x_i} + b) \geq 1 - \xi_i && \forall i \in [m] \\
& \xi_i \geq 0 && \forall i \in [m] \\
& \lambda - \xi_i + (1 - z_i)M \geq 0 && \forall i \in [m] \\
& \xi_i - \lambda + z_i M \geq 0 && \forall i \in [m] \\
& \alpha\lambda - \xi_i + (1 - q_i)M \geq 0 && \forall i \in [m] \\
& \xi_i - \alpha\lambda + q_i M \geq 0 && \forall i \in [m] \\
& p_i \leq q_i && \forall i \in [m] \\
& p_i \leq z_i && \forall i \in [m] \\
& p_i \geq q_i + z_i - 1 && \forall i \in [m] \\
& \xi_i^2 = u_i && \forall i \in [m] \\
& j_i \leq z_i M && \forall i \in [m] \\
& j_i \leq \xi_i && \forall i \in [m] \\
& j_i \geq \xi_i - (1 - z_i)M && \forall i \in [m] \\
& j_i \geq 0 && \forall i \in [m] \\
& k_i \leq q_i M && \forall i \in [m] \\
& k_i \leq u_i && \forall i \in [m] \\
& k_i \geq u_i - (1 - q_i)M && \forall i \in [m] \\
& k_i \geq 0 && \forall i \in [m] \\
& l_i \leq q_i M && \forall i \in [m] \\
& l_i \leq \xi_i && \forall i \in [m] \\
& l_i \geq \xi_i - (1 - q_i)M && \forall i \in [m] \\
& l_i \geq 0 && \forall i \in [m] \\
& \mathbf{w} \in \mathbb{R}^n \qquad \mathbf{p} \in \mathbb{R}^m \\
& b \in \mathbb{R} \qquad \mathbf{u} \in \mathbb{R}^m \\
& \boldsymbol{\xi} \in \mathbb{R}^m \qquad \mathbf{j} \in \mathbb{R}^m \\
& \mathbf{z} \in \{0, 1\}^m \qquad \mathbf{k} \in \mathbb{R}^m \\
& \mathbf{q} \in \{0, 1\}^m \qquad \mathbf{l} \in \mathbb{R}^m
\end{aligned}
\tag{4.21}
$$

This chapter presented eleven models. Three known models from the literature, the benchmark models, and eight new models using either bootstrapping, a capped L_1 cost function or cost functions from the field of feature selection. For all models, it was mentioned whether there is the option of writing them in dual form. Having this possibility is important because the dual form lets us obtain information about the supporting vectors and apply the kernel trick. In the next chapter, all models are tested.

Chapter 5

Results

All models from Chapter 4 are tested, on the two dataset types explained in Chapter 3, for various parameters. In this chapter, we present the most interesting results. Further details can be found Appendix B. All tests are run on thirty datasets with the same parameter values. This results in an average score, which is the percentage of correctly classified points in the test set, the standard deviation of the score and the average time in seconds it took the model to run. The tests are run on a MacBook Pro (15-inch, 2019) with a 2,3 GHz 8-Core Intel Core i9 processor. Since, for some parameter values, the models take a long time to run, a model is considered as not working if, on a single set, the model takes longer than a second to run. Testing for ten different values for a parameter for eleven models on thirty different datasets gives us an upper bound of $10 \times 11 \times 30 \div 60 = 55$ minutes for each test run. First, we analyze what models work in which situations, and then, we go over which models perform better in the cases they do work.

5.1 Runtime

In the first test, see Table 5.1, the number of data points increases while keeping all other values constant. The outcome shows that the SCAD SVM stops working when the dataset is very small. The dataset is so small that we do not consider the model to be useful. The Elastic SCAD SVM ceases to work as well at a small dataset, but the dataset is large enough to still have useful applications. Both models use two binary variables and 22 constraints for every data point; these conditions could explain why the models are slow when the number of points increases. The Elastic SCAD SVM has a L_2 component, possibly allowing for larger datasets than the basic SCAD SVM. The capped L_1 SVM uses a binary variable for every point but only four constraints and a single binary variable. This could explain why the model works on more points than the SCAD SVM and the Elastic SCAD SVM. The method based on bootstrapping solves the L_1 SVM problem several times, so we expect the model to be slower than the L_1 SVM. This prediction is confirmed, and the bootstrapping method breaks at the same number of points as the L_0^b SVM and the capped L_1 SVM. The models use fewer binary variables and constraints

than the SCAD SVMs, and this could be the reason why they are faster than those models. However, they are likely slower than the L_1 SVM and the L_2 SVM because their duals are not obtained. The same holds for the $L_1L_0^b$ SVM and the $L_2L_0^b$ SVM. The L_0^b component in these models slows the models down. The L_0^a SVM uses a single binary variable and has three constraints, and it was expected to be faster than the SCAD SVM, the Elastic SCAD SVM and the capped L_1 SVM. It is also quicker than some models that do not use a binary variable and have less than three constraints, which is surprising. The solver finds the dual for the L_2L_1 SVM, since it exists out of two models with known duals, and thus works with the same number of points as the benchmark models with known duals.

m	5	100	1000	2000	3000	5000
	SCAD	El. SCAD	Boots.	$L_1L_0^b$	L_0^a	L_1
			Cap. L_1	$L_2L_0^b$		L_2
			L_0^b			L_2L_1

Table 5.1: The highest value for m where the model works with D1: $5 \leq m \leq 10000$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. More details in Table B.1 - Table B.4.

The number of features in the dataset is subsequently increased, ceteris paribus. Looking at Table 5.2, it is clear that the SCAD SVM never works. This is expected, since we test here with 100 points and we already saw in Table 5.1 that this model stops working when the dataset contains 5 or fewer points. The Elastic SCAD SVM fails to work at 100 features and all other models continue to work. The Elastic SCAD SVM likely breaks because it already has troubles with datasets larger than 100 points with 5 features and datasets with 100 points and more than 100 features are too large as well. The number of constraints does not increase when the number of features increases, so a large number of features does not cause the other models to stop working.

n	100	2000
	El. SCAD	L_0^a
		L_1
		L_2
		Boots.
		Cap. L_1
		L_0^b
		L_2L_1
		$L_2L_0^b$
		$L_1L_0^b$

Table 5.2: The highest value for n where the model works with D1: $m = 100$, $5 \leq n \leq 2000$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. More details in Table B.5 and Table B.6.

Afterwards, while keeping all other values constant, the share of overlap of the classes in the dataset increases. See Table 5.3. The models with a L_0^b component work less with overlapping classes. The L_0^b cost function is non-convex causing the problem to be NP-hard to solve when some ξ_i become nonzero (Jain and Kar, 2017). The L_0^b SVM immediately stops working, the $L_1L_0^b$ SVM works at the datasets with more overlap and the $L_2L_0^b$ SVM never ceases working. The last model, though, has the highest runtime of all models still performing at $overlap = 1$. The Elastic SCAD SVM quits working as well at a low share of *overlap*, since its cost function is also non-convex.

overlap	0	0.1	0.4	1
	L_0^b	El. SCAD	$L_1L_0^b$	L_0^a
				L_1
				L_2
				Boots.
				Cap. L_1
				L_2L_1
				$L_2L_0^b$

Table 5.3: The highest value for *overlap* where the model works with D1: $m = 100$, $n = 5$, $scale = 10$, $0 \leq overlap \leq 1$, $noise = 0$, $outlier = 0$. More details in Table B.7 and Table B.8.

Similar results are found in Table 5.4. When *noise* increases from zero, some ξ_i become nonzero, and the L_0^b component and the Elastic SCAD SVM stop working.

noise	0.1	0.2	0.5	0.9	1
	L_0^b	El. SCAD	$L_1 L_0^b$	$L_2 L_0^b$	L_0^a L_1 L_2 Boots. Cap. L_1 $L_2 L_1$

Table 5.4: The highest value for *noise* where the model works with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0$, $0 \leq noise \leq 1$, $outlier = 1$. More details in Table B.9 and Table B.10.

In the test that follows the factor *outlier* increases, ceteris paribus. See Table 5.5. The L_0^b SVM and the Elastic SCAD SVM never work, the other two L_0^b component SVMs stop working at some point and the other models continue to work.

outlier	3	10	500
	$L_1 L_0^b$	$L_2 L_0^b$	L_0^a L_1 L_2 Boots. Cap. L_1 $L_2 L_1$

Table 5.5: The highest value for *outlier* where the model works with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0.5$, $1 \leq outlier \leq 500$. More details in Table B.11 and Table B.12.

5.2 Score

5.2.1 Uniform distributed datasets

In the previous section, the SCAD SVM works only when the number of points is low. In this case, see Table 5.6, the model works as well as the L_1 SVM and the L_2 SVM and better than the L_0^a SVM. The score and standard deviation are equal among the last three models in the table since they each find the same solution. This is because there are no misclassifications in the dataset. In the datasets where the Elastic SCAD SVM might be useful, with nonzero $\xi_i's$, the model performs either similarly to the L_1 and L_2 SVMs or does not work. Since the SCAD SVMs either do not work or perform similarly to the benchmark models and since the capped L_1 SVM only performs better in a very specific setting, only the other five new models are considered in the next results.

m	L_0^a SVM	L_1 SVM	L_2 SVM	SCAD SVM
5	0.4000	0.8200	0.8200	0.8200
	0.1742	0.2538	0.2538	0.2538

Table 5.6: Results for benchmark and SCAD SVM with D1: $m = 10$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first and second rows are the score and std. dev. respectively. More details in Table B.1 and Table B.2

The score and standard deviation of all other new models are not affected by the number of data points. See Table 5.7. An increase in the number of data points does not necessarily lead to higher model scores or reduced deviation. The same holds for the L_1 SVM and the L_2 SVM. In contrast, the L_0^a SVM scores higher with more data points.

m	Boots. SVM	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM
10	0.8300	0.8167	0.8233	0.8233	0.8233
	0.2136	0.2135	0.2063	0.2063	0.2063
1000	0.8537	0.8495	0.8500	0.8500	0.8500
	0.1914	0.1894	0.1838	0.1838	0.1838

Table 5.7: Results for five SVMs with D1: $10 \leq m \leq 1000$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first and second rows are the score and std. dev. respectively. More details in Table B.1 - Table B.4

For all models, it holds that the score becomes higher and the standard deviation becomes lower with an increased number of features. See Table 5.8.

n	Boots. SVM	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM
5	0.8870	0.9263	0.9140	0.9140	0.9140
	0.1550	0.1432	0.1482	0.1482	0.1482
200	1.0000	1.0000	1.0000	1.0000	1.0000
	0.0000	0.0000	0.0000	0.0000	0.0000

Table 5.8: Results for five SVMs with D1: $5 \leq n \leq 200$, $m = 100$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first and second rows are the score and std. dev. respectively. More details in Table B.5 and Table B.6

In the previous section, we found that the L_0^b SVM, the $L_2L_0^b$ SVM and the $L_1L_0^b$ SVM work less with nonzero ξ_i's. For that reason, we do not consider these models when comparing results for nonzero *overlap*, *noise*, and *outlier*. When the data has overlapping classes and no noise, the method based on bootstrapping performs worse than the L_1 SVM and the L_2 SVM and better than the L_0^a SVM, see Table 5.9. The L_2L_1 SVM has a score between the scores of the L_1 SVM and the L_2 SVM; a coherent result since it uses both

components of both models.

overlap	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	L_2L_1 SVM
0.3	0.4760	0.5810	0.5857	0.5543	0.5810
	0.0240	0.1106	0.1037	0.0871	0.1015

Table 5.9: Results for benchmark, bootstrapping and L_2L_1 SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0.3$, $noise = 0$, $outlier = 0$. The first and second rows are the score and std. dev. respectively. More details in Table B.7 and Table B.8

With noise and without overlap and outliers in the dataset, see Table 5.10, the method based on bootstrapping performs worse than the L_1 SVM and the L_2 SVM. The L_2L_1 SVM performs similarly to the L_1 SVM and the L_2 SVM.

noise	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	L_2L_1 SVM
0.5	0.5283	0.7667	0.7783	0.6720	0.7783
	0.1402	0.2183	0.2082	0.1996	0.2083

Table 5.10: Results for benchmark, bootstrapping and L_2L_1 SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0.5$, $outlier = 1$. The first and second rows are the score and std. dev. respectively. More details in Table B.9 and Table B.10

Table 5.11 shows that when outliers are added to the dataset, the method based on bootstrapping performs better than all investigated models, when the outliers are not too large. When the outliers get very large, the capped L_1 SVM achieves the highest scores of all models. As mentioned earlier, this is the only situation where this model performs better than the others. We conclude that with medium outliers the method based on bootstrapping performs best and with large outliers the capped L_1 SVM performs best. The L_2L_1 SVM performs again similarly to the L_1 SVM and the L_2 SVM.

outlier	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
10	0.5590	0.6047	0.6917	0.7067	0.6000
	0.1855	0.2089	0.2161	0.1950	0.2015
500	0.4833	0.4833	0.4807	0.5453	0.6143
	0.0294	0.0294	0.0277	0.1213	0.2156

Table 5.11: Results for benchmark, bootstrapping and capped L_1 SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0.3$, $noise = 0.5$, $10 \leq outlier \leq 500$. The first and second rows are the score and std. dev. respectively. More details in Table B.11 and Table B.12

5.2.2 Normal distributed datasets

So far, uniform distributed datasets have been used. Now, we investigate if similar results are found using normal distributions. The dataset used to generate Table 5.12 has overlapping classes, and an outcome identical to one found in Table 5.9 is expected. Indeed the method based on bootstrapping performs the worst and the L_2L_1 SVM performs similarly to the L_1 SVM and the L_2 SVM.

σ	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	L_2L_1 SVM
20	0.8743	0.8137	0.8137	0.7473	0.8137
	0.0362	0.0492	0.0492	0.0640	0.0492

Table 5.12: Results for benchmark, bootstrapping and L_2L_1 SVMs with D2: $m = 100$, $n = 30$, features class 1: $N(10, 20)$, features class 2: $N(20, 20)$, no noise. The first and second rows are the score and std. dev. respectively. More details in Table B.13 and Table B.14

The dataset used for Table 5.13 has noise and medium outliers, similar to the first row in Table 5.11. Once more, the method based on bootstrapping performs the best and the L_2L_1 SVM has a score between the scores of the L_1 SVM and the L_2 SVM.

$noise1, 2$	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	L_2L_1 SVM
0.5	0.6467	0.7583	0.7493	0.8873	0.7473
	0.1559	0.0945	0.1023	0.1010	0.1073

Table 5.13: Results for benchmark, bootstrapping and L_2L_1 SVMs with D2: $m = 100$, $n = 30$, features class 1: $N(10, 20)$, features class 2: $N(20, 20)$, noise: $N(0, 100)$, $noise1 = noise2 = 0.5$. The first and second rows are the score and std. dev. respectively. More details in Table B.15 and Table B.16

In this chapter, we presented the results for all the models. The benchmark models, the method based on bootstrapping, the capped L_1 SVM and the L_2L_1 SVM perform well when examining the runtime. The method based on bootstrapping performs well in datasets with medium outliers and noise, the capped L_1 SVM has a higher score in datasets with large outliers and noise, in all other cases the L_1 SVM, the L_2 SVM and the L_2L_1 SVM score better.

Chapter 6

Conclusion

The outcome of our research shows that most functions used in feature selection in the context they were presented cannot deal with misclassification in a support vector machine as well as the already known models can. There is the possibility that the SCAD and the Elastic SCAD perform better when using specialized algorithms that lower the runtime of these models. The only exception is the L_2L_1 SVM, which achieves scores similar to those of the benchmark models. We found that the method based on bootstrapping works better than the regular models when datasets have medium outliers and noise. The method performs worse when datasets have overlapping classes and noise but no outliers. Since this method uses the L_1 SVM, it does not lose the ability to deal with nonlinear separating boundaries. Information about the supporting vectors can still be obtained. The capped L_1 performs better on datasets with large outliers. This model cannot use nonlinear boundaries and does not allow us to obtain information about the support vectors.

The results show that the L_1 and L_2 cost functions should be used in most situations when using an SVM. When datasets have noise and outliers or are expected to have noise and outliers, using bootstrapping is recommended. In cases where bootstrapping does not work because the outliers are too large, use the capped L_1 SVM. Feature selection penalty functions are not advised to be used as misclassification cost functions, without specialized algorithms, since they are outperformed by the L_1 and L_2 cost functions.

For future work, we suggest further research into the application of bootstrapping in support vector machines. This thesis uses only the L_1 cost function in the method based on bootstrapping. Although, the L_2 cost function could perform better. We used ideas from the field of feature selection, not feature selection itself. It is worth investigating the performance of bootstrapping when feature selection is applied. Otherwise, when using specialized algorithms to decrease the runtime of the L_0^b and the SCAD models, repeating the tests in this thesis and checking whether the models that now were not fast enough get higher results than the benchmark models could be an interesting approach.

Bibliography

Abe, S. (2005). *Support vector machines for pattern classification*, volume 2. Springer.

Al-Thanoon, N. A., Qasim, O. S., and Algamal, Z. Y. (2018). Tuning parameter estimation in scad-support vector machine using firefly algorithm with application in gene selection and cancer classification. *Computers in Biology and Medicine*, 103:262–268.

Becker, N., Toedt, G., Lichter, P., and Benner, A. (2011). Elastic scad as a novel penalization method for svm classification tasks in high-dimensional data. *BMC Bioinformatics*.

Borah, P. and Gupta, D. (2017). Review: Support vector machines in pattern recognition. *International Journal of Engineering and Technology*, 9(3).

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA. Association for Computing Machinery.

Bousquet, O., von Luxburg, U., and Rätsch, G. (2003). Advanced lectures on machine learning. Technical report, ML Summer Schools.

Boyle, B. H. (2011). *Support vector machines: data analysis, machine learning and applications*. Nova Science Publishers, Inc., Switzerland.

Cortes, C. and Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*, (20):272–297.

Fan, J. (1997). Comments on «wavelets in statistics: A review» by a. antoniadis. *J. Ital. Statist. Soc.*

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of American Statistical Association*.

Hoerl, A. and Kennard, R. (1979). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1).

Jain, P. and Kar, P. (2017). Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363.

Kecman, V. (2004). Support vector machines basics. Technical report, The University of Auckland.

Li, H.-X., Yang, J.-L., Zhang, G., and Fan, B. (2013). Probabilistic support vector machines for classification of noise affected data. *Information Sciences*, 221:60–71.

Lin, C.-J. (2001). Formulations of support vector machines: A note from an optimization point of view. *Neural Computation*, (13):307–317.

Mangasarian, O. and Bradley, P. (1998). Feature selection via concave minimization and support vector machines. Technical report, University of Wisconsin.

Murty, M. and Raghava, R. (2016). *Support Vector Machines and Perceptrons*. SpringerBriefs, Switzerland.

Neumann, J., Schnörr, C., and Steidl, G. (2005). Combined svm-based feature selection and classification. *Mach Learn*.

Scholkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. The MIT Press.

Tibishirani, R. (1996). Regression shrinkage and selection via the lasso. *ournal of the Royal Statistical Society: Series B (Methodological)*.

Vapnik, V. N. and Chervonenkis, A. Y. (1964a). A class of algorithms for pattern recognition learning. *Avtomatika i Telemekhanika*, 25(6):937–945.

Vapnik, V. N. and Chervonenkis, A. Y. (1964b). On a perceptron class. *Avtomatika i Telemekhanika*, 25(1):112–120.

Wong, T.-T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48(9):2839–2846.

Yu, W., Liu, T., Valdez, R., Gwinn, M., and Khoury, M. J. (2010). Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes. *BMC Medical Informatics and Decision Making*, 10(16).

Zhang, H. H., Ahn, J., Lin, X., and Park, C. (2006). Gene selection using support vector machines with non-convex penalty. *Bioinformatics*.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*.

Appendix A

Cost function plots

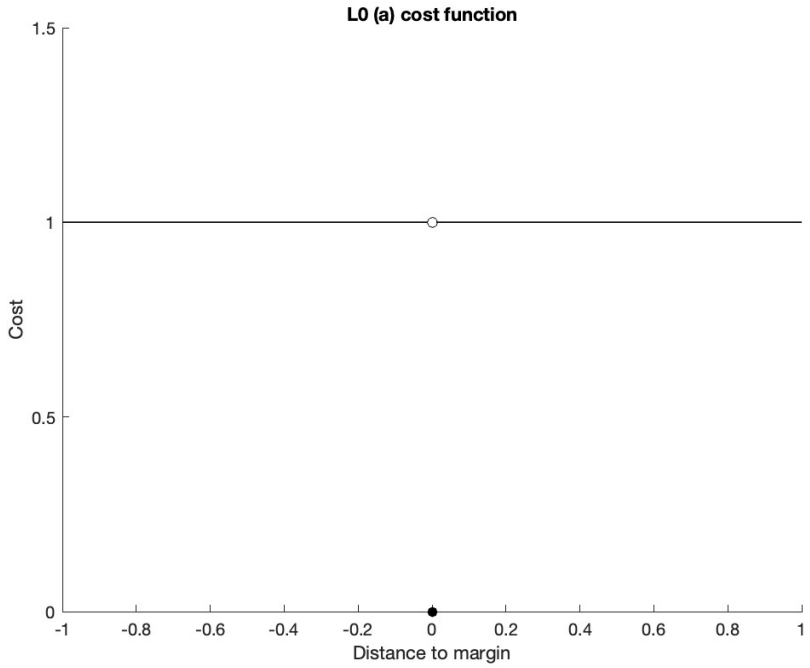

Figure A.1: Cost function of the L_0^a SVM. Source: Wim van Duuren.

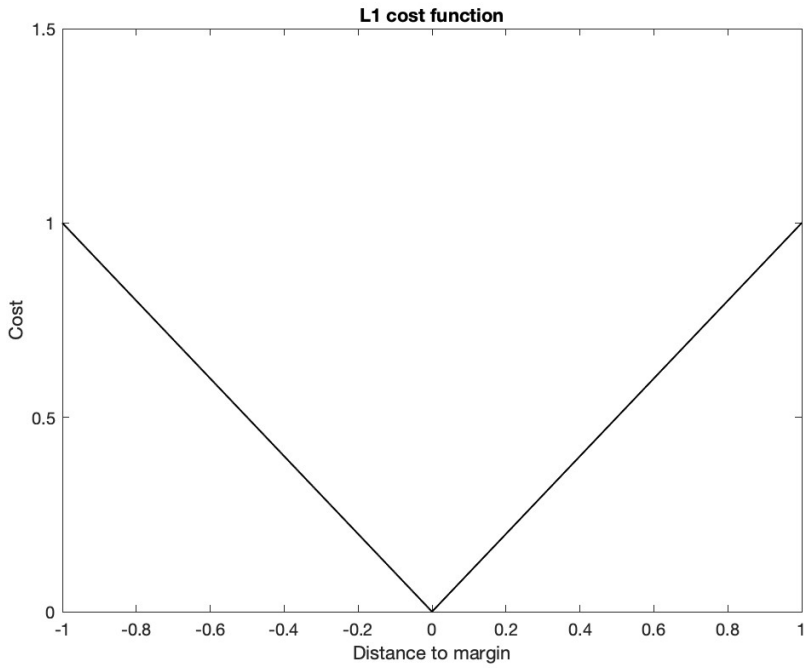

Figure A.2: Cost function of the L_1 SVM. Source: Wim van Duuren.

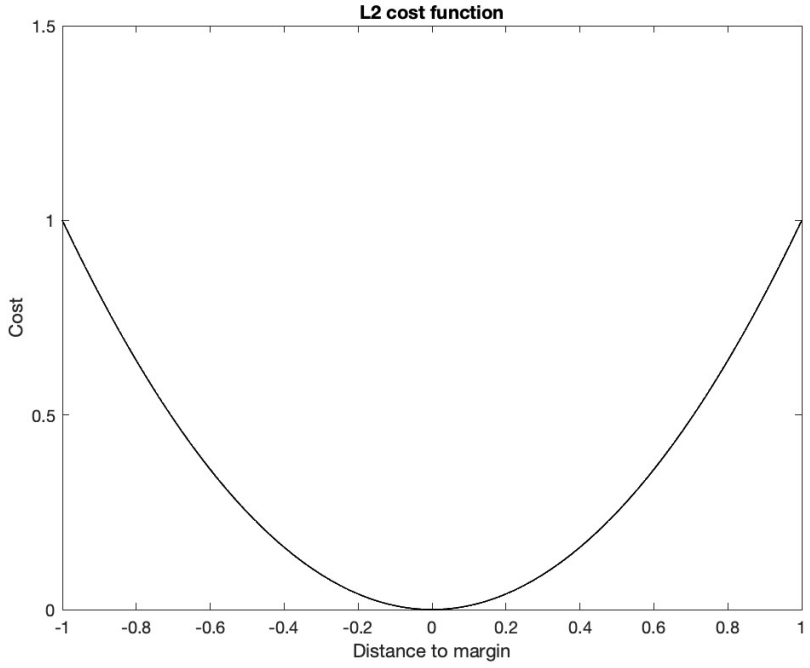

Figure A.3: Cost function of the L_2 SVM. Source: Wim van Duuren.

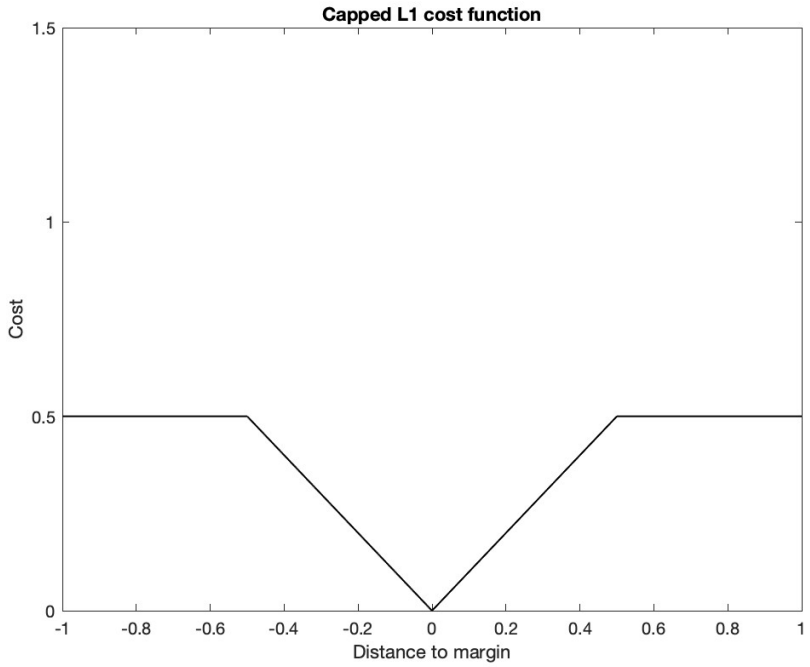

Figure A.4: Cost function of the capped L_1 SVM with $d = 0.5$. Source: Wim van Duuren.

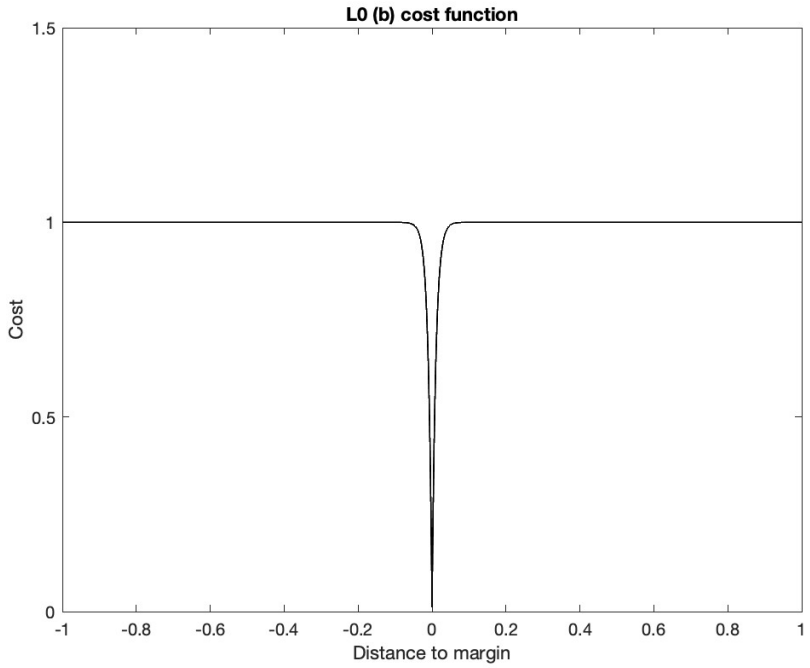

Figure A.5: Cost function of the L_0^b SVM with $\alpha = 100$. Source: Wim van Duuren.

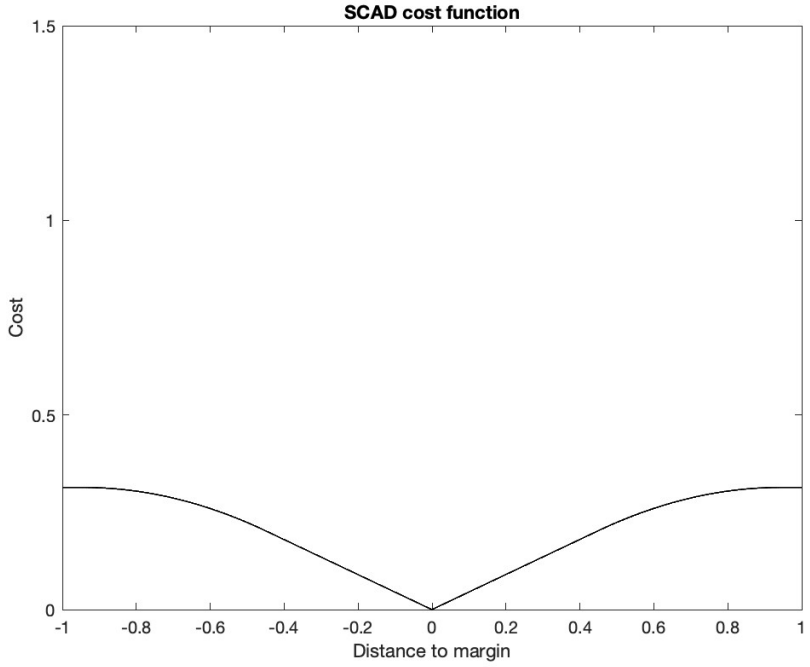

Figure A.6: Cost function of the SCAD SVM with $\alpha = 2.1$ and $\lambda = 0.45$. Source: Wim van Duuren.

Appendix B

Detailed results

m	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
5	0.4000	0.8200	0.8200	0.7600	0.4200
	0.1742	0.2538	0.2538	0.2848	0.1846
	0.0069	0.0042	0.0043	0.0221	0.0077
10	0.4500	0.8233	0.8233	0.8300	0.4367
	0.1333	0.2063	0.2063	0.2136	0.1273
	0.0065	0.0042	0.0045	0.0216	0.0074
20	0.4900	0.8217	0.8217	0.8133	0.6017
	0.0662	0.1994	0.1994	0.2072	0.2534
	0.0061	0.0045	0.0042	0.0192	0.0068
30	0.4778	0.7644	0.7644	0.8089	0.6467
	0.0563	0.2071	0.2071	0.1904	0.2727
	0.0059	0.0042	0.0041	0.0196	0.0072
50	0.5300	0.8593	0.8593	0.8533	0.6627
	0.1486	0.1645	0.1645	0.1851	0.2347
	0.0075	0.0055	0.0046	0.0217	0.0089
100	0.7893	0.8270	0.8270	0.8150	0.6983
	0.2119	0.1975	0.1975	0.1878	0.2246
	0.0098	0.0107	0.0096	0.0269	0.0103
200	0.8097	0.8162	0.8162	0.8073	0.6963
	0.2003	0.1997	0.1997	0.1819	0.2589
	0.0161	0.0247	0.0324	0.0379	0.0192
300	0.8300	0.8207	0.8207	0.8134	0.6837
	0.2140	0.2158	0.2158	0.2227	0.2139
	0.0181	0.0520	0.0406	0.0605	0.0377

Table B.1: Results first five SVMs with D1: $5 \leq m \leq 300$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first, second and third row are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

m	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
5	0.8333	0.8200	0.8200	0.8200	0.8200	0.8200
	0.2578	0.2538	0.2538	0.2538	0.2538	0.2538
	0.0107	0.0049	0.0095	0.0088	0.0847	0.0159
10	0.8167	0.8233	0.8233	0.8233	0	0.8200
	0.2135	0.2063	0.2063	0.2063	0	0.2091
	0.0113	0.0048	0.0084	0.0079	0	0.0296
20	0.8150	0.8217	0.8217	0.8217	0	0.8233
	0.1970	0.1994	0.1994	0.1994	0	0.1982
	0.0095	0.0050	0.0077	0.0072	0	0.0416
30	0.8000	0.7644	0.7644	0.7644	0	0.7656
	0.2011	0.2071	0.2071	0.2071	0	0.2080
	0.0106	0.0052	0.0089	0.0077	0	0.0550
50	0.8620	0.8593	0.8593	0.8593	0	0.8593
	0.1752	0.1645	0.1645	0.1645	0	0.1645
	0.0130	0.0054	0.0096	0.0093	0	0.0787
100	0.8253	0.8270	0.8270	0.8270	0	0.8270
	0.1972	0.1975	0.1975	0.1975	0	0.1975
	0.0181	0.0096	0.0129	0.0127	0	0.2216
200	0.8267	0.8162	0.8162	0.8162	0	0
	0.1954	0.1997	0.1997	0.1997	0	0
	0.0364	0.0238	0.0224	0.0222	0	0
300	0.8387	0.8207	0.8207	0.8207	0	0
	0.2079	0.2158	0.2158	0.2158	0	0
	0.0702	0.0359	0.0360	0.0361	0	0

Table B.2: Results last six SVMs with D1: $5 \leq m \leq 300$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

m	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
500	0.8475	0.8291	0.8291	0.8357	0.6689
	0.1844	0.1949	0.1949	0.1888	0.2177
	0.0387	0.1036	0.0997	0.1415	0.0984
1000	0.8499	0.8500	0.8500	0.8537	0.7426
	0.1940	0.1838	0.1838	0.1914	0.2211
	0.0963	0.0250	0.0264	0.3922	0.2193
2000	0.8777	0.8561	0.8561	0	0
	0.1648	0.1850	0.1850	0	0
	0.2462	0.0610	0.0715	0	0
3000	0.7911	0.8103	0.8103	0	0
	0.1787	0.1885	0.1885	0	0
	0.4908	0.1220	0.1484	0	0
5000	0	0.8377	0.8377	0	0
	0	0.1981	0.1981	0	0
	0	0.2739	0.3329	0	0
10000	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0

Table B.3: Results first five SVMs with D1: $500 \leq m \leq 10000$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

m	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
500	0.8454	0.8291	0.8291	0.8291	0	0
	0.1864	0.1949	0.1949	0.1949	0	0
	0.1758	0.0874	0.0835	0.0833	0	0
1000	0.8495	0.8500	0.8500	0.8500	0	0
	0.1894	0.1838	0.1838	0.1838	0	0
	0.5721	0.0242	0.2328	0.2239	0	0
2000	0	0.8561	0.8561	0.8561	0	0
	0	0.1850	0.1850	0.1850	0	0
	0	0.0704	0.6871	0.6640	0	0
3000	0	0.8103	0	0	0	0
	0	0.1885	0	0	0	0
	0	0.1437	0	0	0	0
5000	0	0.8377	0	0	0	0
	0	0.1981	0	0	0	0
	0	0.3262	0	0	0	0
10000	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0

Table B.4: Results last six SVMs with D1: $500 \leq m \leq 1000$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

n	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
5	0.8680	0.9140	0.9140	0.8870	0.7063
	0.2006	0.1482	0.1482	0.1550	0.2368
	0.0135	0.0141	0.0121	0.0328	0.0137
10	0.9193	0.9117	0.9117	0.9190	0.8293
	0.1746	0.1737	0.1737	0.1603	0.2192
	0.0143	0.0128	0.0127	0.0350	0.0123
20	0.9640	0.9653	0.9653	0.9633	0.7993
	0.1192	0.1178	0.1178	0.1191	0.2495
	0.0151	0.0170	0.0165	0.0375	0.0157
30	0.9997	0.9997	0.9997	0.9987	0.7343
	0.0018	0.0018	0.0018	0.0057	0.2629
	0.0147	0.0165	0.0200	0.0398	0.0192
50	1.0000	0.9990	0.9990	0.9953	0.7567
	0.0000	0.0055	0.0055	0.0256	0.2479
	0.0167	0.0228	0.0221	0.0403	0.0202
100	1.0000	1.0000	1.0000	1.0000	0.6917
	0.0000	0.0000	0.0000	0.0000	0.2621
	0.0227	0.0440	0.0339	0.0430	0.0282
200	1.0000	1.0000	1.0000	1.0000	0.7363
	0.0000	0.0000	0.0000	0.0000	0.2606
	0.0338	0.0533	0.0428	0.0600	0.0363
300	1.0000	1.0000	1.0000	1.0000	0.7753
	0.0000	0.0000	0.0000	0.0000	0.2601
	0.0514	0.0609	0.0607	0.0888	0.0514
500	1.0000	1.0000	1.0000	1.0000	0.6620
	0.0000	0.0000	0.0000	0.0000	0.2571
	0.0847	0.0771	0.0638	0.2164	0.0693
1000	1.0000	1.0000	1.0000	1.0000	0.6740
	0.0000	0.0000	0.0000	0.0000	0.2504
	0.1638	0.1205	0.1108	0.2359	0.1300
2000	1.0000	1.0000	1.0000	1.0000	0.6157
	0.0000	0.0000	0.0000	0.0000	0.2305
	0.3867	0.2453	0.2396	0.6345	0.2466

Table B.5: Results first five SVMs with D1: $m = 100$, $5 \leq n \leq 2000$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

n	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
5	0.9263	0.9140	0.9140	0.9140	0	0.9140
	0.1432	0.1482	0.1482	0.1482	0	0.1482
	0.0206	0.0122	0.0161	0.0140	0	0.2745
10	0.9133	0.9117	0.9117	0.9117	0	0.9117
	0.1698	0.1737	0.1737	0.1737	0	0.1737
	0.0193	0.0137	0.0169	0.0154	0	0.3177
20	0.9657	0.9653	0.9653	0.9653	0	0.9653
	0.1179	0.1178	0.1178	0.1178	0	0.1178
	0.0223	0.0145	0.0204	0.0176	0	0.3807
30	0.9997	0.9997	0.9997	0.9997	0	0.9997
	0.0018	0.0018	0.0018	0.0018	0	0.0018
	0.0254	0.0162	0.0199	0.0185	0	0.4159
50	0.9990	0.9990	0.9990	0.9990	0	0.9987
	0.0055	0.0055	0.0055	0.0055	0	0.0073
	0.0272	0.0204	0.0207	0.0209	0	0.4675
100	1.0000	1.0000	1.0000	1.0000	0	1.0000
	0.0000	0.0000	0.0000	0.0000	0	0.0000
	0.0323	0.0322	0.0251	0.0256	0	0.7552
200	1.0000	1.0000	1.0000	1.0000	0	0
	0.0000	0.0000	0.0000	0.0000	0	0
	0.0483	0.0428	0.0351	0.0340	0	0
300	1.0000	1.0000	1.0000	1.0000	0	0
	0.0000	0.0000	0.0000	0.0000	0	0
	0.0608	0.0591	0.0391	0.0421	0	0
500	1.0000	1.0000	1.0000	1.0000	0	0
	0.0000	0.0000	0.0000	0.0000	0	0
	0.1041	0.0674	0.0810	0.0759	0	0
1000	1.0000	1.0000	1.0000	1.0000	0	0
	0.0000	0.0000	0.0000	0.0000	0	0
	0.2173	0.0990	0.1339	0.1440	0	0
2000	1.0000	1.0000	1.0000	1.0000	0	0
	0.0000	0.0000	0.0000	0.0000	0	0
	0.5980	0.2272	0.3505	0.3679	0	0

Table B.6: Results last six SVMs with D1: $m = 100$, $5 \leq n \leq 2000$, $scale = 10$, $overlap = 0$, $noise = 0$, $outlier = 0$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

overlap	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
0	0.8653	0.8583	0.8583	0.8430	0.6607
	0.1950	0.1782	0.1782	0.1830	0.2573
	0.0123	0.0116	0.0114	0.0344	0.0121
0.1	0.4757	0.5683	0.5743	0.5583	0.5540
	0.0273	0.1292	0.1384	0.0944	0.1340
	0.0136	0.0105	0.0117	0.0369	0.0127
0.2	0.4717	0.5843	0.5990	0.6040	0.5290
	0.0251	0.1187	0.1340	0.1287	0.1242
	0.0100	0.0083	0.0097	0.0314	0.0116
0.3	0.4760	0.5810	0.5857	0.5543	0.5140
	0.0240	0.1106	0.1037	0.0871	0.0929
	0.0100	0.0088	0.0095	0.0317	0.0117
0.4	0.4710	0.6137	0.6120	0.5817	0.5087
	0.0231	0.1035	0.1036	0.0969	0.0476
	0.0105	0.0096	0.0114	0.0329	0.0112
0.5	0.4780	0.5450	0.5503	0.5157	0.5003
	0.0185	0.0811	0.0897	0.0529	0.0480
	0.0116	0.0097	0.0113	0.0327	0.0127
0.6	0.4733	0.5237	0.5183	0.5220	0.5053
	0.0275	0.0595	0.0545	0.0520	0.0505
	0.0122	0.0104	0.0102	0.0343	0.0129
0.7	0.4810	0.5293	0.5273	0.5130	0.4977
	0.0243	0.0547	0.0482	0.0383	0.0404
	0.0118	0.0101	0.0090	0.0347	0.0122
0.8	0.4693	0.5017	0.4943	0.5010	0.4943
	0.0274	0.0502	0.0459	0.0454	0.0430
	0.0113	0.0092	0.0088	0.0345	0.0121
0.9	0.4757	0.4987	0.5193	0.5063	0.4967
	0.0231	0.0375	0.0383	0.0390	0.0429
	0.0110	0.0099	0.0088	0.0317	0.0119
1	0.4763	0.4977	0.4990	0.5040	0.5120
	0.0251	0.0396	0.0344	0.0388	0.0357
	0.0112	0.0090	0.0086	0.0330	0.0106

Table B.7: Results first five SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $0 \leq overlap \leq 1$, $noise = 0$, $outlier = 0$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

overlap	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
0	0.8493	0.8583	0.8583	0.8583	0	0.8587
	0.1883	0.1782	0.1782	0.1782	0	0.1783
	0.0213	0.0099	0.0161	0.0148	0	0.2822
0.1	0	0.5767	0.5743	0.5580	0	0.5760
	0	0.1379	0.1384	0.1098	0	0.1420
	0	0.0118	0.0950	0.0981	0	0.6028
0.2	0	0.5967	0.5990	0.5807	0	0
	0	0.1313	0.1340	0.1199	0	0
	0	0.0091	0.1424	0.1485	0	0
0.3	0	0.5810	0.5857	0.5887	0	0
	0	0.1015	0.1037	0.1131	0	0
	0	0.0086	0.1718	0.2138	0	0
0.4	0	0.6113	0.6120	0.6240	0	0
	0	0.1061	0.1036	0.1083	0	0
	0	0.0097	0.2411	0.3216	0	0
0.5	0	0.5497	0.5503	0	0	0
	0	0.0881	0.0897	0	0	0
	0	0.0098	0.2482	0	0	0
0.6	0	0.5183	0.5183	0	0	0
	0	0.0546	0.0545	0	0	0
	0	0.0107	0.2732	0	0	0
0.7	0	0.5277	0.5273	0	0	0
	0	0.0492	0.0482	0	0	0
	0	0.0101	0.3393	0	0	0
0.8	0	0.4933	0.4943	0	0	0
	0	0.0452	0.0459	0	0	0
	0	0.0094	0.3795	0	0	0
0.9	0	0.5197	0.5193	0	0	0
	0	0.0388	0.0383	0	0	0
	0	0.0085	0.4390	0	0	0
1	0	0.4987	0.4990	0	0	0
	0	0.0335	0.0344	0	0	0
	0	0.0085	0.4668	0	0	0

Table B.8: Results last six SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $0 \leq overlap \leq 1$, $noise = 0$, $outlier = 0$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

noise	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
0	0.8050	0.8007	0.8007	0.8297	0.7077
	0.2228	0.1928	0.1928	0.1787	0.2266
	0.0106	0.0115	0.0128	0.0322	0.0125
0.1	0.8550	0.8077	0.8407	0.7783	0.7070
	0.1948	0.1707	0.1744	0.1973	0.2290
	0.0101	0.0100	0.0109	0.0325	0.0114
0.2	0.7510	0.8267	0.8390	0.6753	0.6753
	0.2084	0.1728	0.1822	0.1859	0.2202
	0.0105	0.0087	0.0104	0.0308	0.0114
0.3	0.7160	0.8260	0.8453	0.6887	0.6293
	0.2413	0.2033	0.1970	0.1838	0.1818
	0.0104	0.0098	0.0097	0.0304	0.0116
0.4	0.6043	0.7707	0.7990	0.7870	0.6387
	0.2045	0.1935	0.2087	0.1646	0.2490
	0.0100	0.0105	0.0109	0.0304	0.0124
0.5	0.5283	0.7667	0.7783	0.6720	0.6263
	0.1402	0.2183	0.2082	0.1996	0.1992
	0.0112	0.0093	0.0098	0.0310	0.0122
0.6	0.4933	0.6947	0.7267	0.6617	0.5307
	0.0636	0.2149	0.2179	0.1645	0.2145
	0.0101	0.0088	0.0097	0.0270	0.0113
0.7	0.4727	0.7510	0.7677	0.6930	0.6013
	0.0226	0.2204	0.2185	0.1876	0.1697
	0.0114	0.0088	0.0097	0.0274	0.0105
0.8	0.4783	0.7410	0.7360	0.6900	0.6203
	0.0191	0.1846	0.2035	0.1853	0.2226
	0.0117	0.0088	0.0088	0.0270	0.0107
0.9	0.4800	0.6297	0.6577	0.6573	0.5690
	0.0252	0.2011	0.1974	0.1868	0.2250
	0.0119	0.0083	0.0087	0.0293	0.0104
1	0.4750	0.5370	0.5587	0.4900	0.4713
	0.0257	0.1549	0.1762	0.2533	0.2011
	0.0106	0.0096	0.0100	0.0313	0.0122

Table B.9: Results first five SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0$, $0 \leq noise \leq 1$, $outlier = 1$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

noise	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
0	0.8413	0.8007	0.8007	0.8007	0	0.8007
	0.1906	0.1928	0.1928	0.1928	0	0.1928
	0.0196	0.0097	0.0141	0.0139	0	0.2652
0.1	0.8060	0.8460	0.8407	0.8090	0	0.8350
	0.1918	0.1751	0.1744	0.1755	0	0.1668
	0.0875	0.0095	0.0560	0.0472	0	0.4587
0.2	0	0.8343	0.8390	0.8157	0	0.8427
	0	0.1861	0.1822	0.1779	0	0.1728
	0	0.0078	0.1095	0.1134	0	0.6280
0.3	0	0.8383	0.8453	0.7927	0	0
	0	0.2008	0.1970	0.1993	0	0
	0	0.0105	0.1456	0.1566	0	0
0.4	0	0.7960	0.7990	0.7490	0	0
	0	0.2045	0.2087	0.1893	0	0
	0	0.0134	0.1778	0.2017	0	0
0.5	0	0.7783	0.7783	0.7517	0	0
	0	0.2083	0.2082	0.2096	0	0
	0	0.0102	0.2073	0.2874	0	0
0.6	0	0.7230	0.7267	0	0	0
	0	0.2182	0.2179	0	0	0
	0	0.0090	0.2216	0	0	0
0.7	0	0.7653	0.7523	0	0	0
	0	0.2194	0.2185	0	0	0
	0	0.0098	0.2343	0	0	0
0.8	0	0.7353	0.7360	0	0	0
	0	0.2030	0.2035	0	0	0
	0	0.0090	0.2874	0	0	0
0.9	0	0.6587	0.6577	0	0	0
	0	0.1976	0.1974	0	0	0
	0	0.0080	0.3731	0	0	0
1	0	0.5590	0	0	0	0
	0	0.1768	0	0	0	0
	0	0.0092	0	0	0	0

Table B.10: Results last six SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0$, $0 \leq noise \leq 1$, $outlier = 1$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

outlier	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
1	0.5247	0.7950	0.8520	0.7273	0.6233
	0.1540	0.1937	0.1923	0.2219	0.1984
	0.0105	0.0099	0.0113	0.0306	0.0118
2	0.5230	0.7807	0.7757	0.6740	0.6000
	0.1537	0.2270	0.2370	0.2087	0.2040
	0.0108	0.0111	0.0102	0.0315	0.0120
3	0.5523	0.7847	0.7647	0.7390	0.6470
	0.1850	0.1899	0.1955	0.1965	0.2431
	0.0113	0.0103	0.0110	0.0326	0.0118
5	0.5497	0.7370	0.7463	0.7610	0.6500
	0.1789	0.2006	0.1825	0.2223	0.2015
	0.0114	0.0097	0.0105	0.0308	0.0112
10	0.5590	0.6047	0.6917	0.7067	0.6000
	0.1855	0.2089	0.2161	0.1950	0.2318
	0.0116	0.0104	0.0108	0.0303	0.0119
20	0.4877	0.4917	0.5090	0.5313	0.6020
	0.0732	0.0940	0.1345	0.0935	0.1871
	0.0125	0.0111	0.0112	0.0324	0.0113
30	0.4970	0.4970	0.5313	0.6960	0.6297
	0.0911	0.0911	0.1291	0.2020	0.2028
	0.0128	0.0140	0.0100	0.0306	0.0118
50	0.4593	0.4593	0.4817	0.5783	0.5773
	0.0745	0.0745	0.0928	0.1575	0.1670
	0.0123	0.0134	0.0109	0.0281	0.0113
100	0.4837	0.4837	0.4640	0.5573	0.5927
	0.1085	0.1085	0.0856	0.1433	0.1844
	0.0124	0.0152	0.0121	0.0286	0.0115
200	0.4703	0.4703	0.4703	0.5730	0.5740
	0.0254	0.0254	0.0254	0.1359	0.2319
	0.0120	0.0122	0.0108	0.0299	0.0113
300	0.4720	0.4720	0.4607	0.5917	0.6340
	0.0243	0.0243	0.0691	0.1806	0.1891
	0.0119	0.0119	0.0119	0.0278	0.0112
500	0.4833	0.4833	0.4807	0.5453	0.6143
	0.0294	0.0294	0.0277	0.1213	0.2156
	0.0121	0.0141	0.0122	0.0288	0.0119

Table B.11: Results first five SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0.5$, $1 \leq outlier \leq 500$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

outlier	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
1	0	0.8497	0.8520	0.7833	0	0
	0	0.1935	0.1923	0.2089	0	0
	0	0.0108	0.1784	0.2133	0	0
2	0	0.7780	0.7757	0.7850	0	0
	0	0.2362	0.2370	0.2236	0	0
	0	0.0105	0.1990	0.2246	0	0
3	0	0.7690	0.7647	0.7893	0	0
	0	0.1963	0.1955	0.1934	0	0
	0	0.0108	0.2029	0.2767	0	0
5	0	0.7503	0.7500	0	0	0
	0	0.1833	0.1780	0	0	0
	0	0.0092	0.2146	0	0	0
10	0	0.6880	0.6587	0	0	0
	0	0.2161	0.2227	0	0	0
	0	0.0093	0.2729	0	0	0
20	0	0.5090	0	0	0	0
	0	0.1345	0	0	0	0
	0	0.0109	0	0	0	0
30	0	0.5267	0	0	0	0
	0	0.1187	0	0	0	0
	0	0.0105	0	0	0	0
50	0	0.4793	0	0	0	0
	0	0.0978	0	0	0	0
	0	0.0100	0	0	0	0
100	0	0.4693	0	0	0	0
	0	0.0573	0	0	0	0
	0	0.0112	0	0	0	0
200	0	0.4703	0	0	0	0
	0	0.0254	0	0	0	0
	0	0.0092	0	0	0	0
300	0	0.4723	0	0	0	0
	0	0.0253	0	0	0	0
	0	0.0095	0	0	0	0
500	0	0.4807	0	0	0	0
	0	0.0277	0	0	0	0
	0	0.0096	0	0	0	0

Table B.12: Results last six SVMs with D1: $m = 100$, $n = 5$, $scale = 10$, $overlap = 0$, $noise = 0.5$, $1 \leq outlier \leq 500$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

σ	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
0	0.9950	0.9950	0.9950	0.9950	0.5003
	0.0051	0.0051	0.0051	0.0051	0.0418
	0.0083	0.0139	0.0156	0.0361	0.0117
1	0.9953	0.9947	0.9947	0.9953	0.4627
	0.0051	0.0051	0.0051	0.0051	0.0245
	0.0126	0.0184	0.0197	0.0388	0.0163
2	0.9947	0.9937	0.9937	0.9947	0.4910
	0.0051	0.0049	0.0049	0.0051	0.0812
	0.0136	0.0169	0.0208	0.0382	0.0174
5	0.9963	0.9940	0.9940	0.9953	0.5293
	0.0049	0.0056	0.0056	0.0057	0.1069
	0.0160	0.0199	0.0231	0.0395	0.0197
10	0.9883	0.9747	0.9747	0.9663	0.5610
	0.0105	0.0172	0.0172	0.0270	0.0912
	0.0191	0.0224	0.0211	0.0403	0.0200
20	0.8743	0.8137	0.8137	0.7473	0.5530
	0.0362	0.0492	0.0492	0.0640	0.0780
	0.0180	0.0192	0.0226	0.0375	0.0189
50	0.6043	0.5937	0.5993	0.5557	0.5117
	0.0474	0.0659	0.0635	0.0473	0.0579
	0.0147	0.0133	0.0144	0.0345	0.0138
100	0.5390	0.5303	0.5373	0.5237	0.5017
	0.0535	0.0565	0.0563	0.0595	0.0601
	0.0147	0.0131	0.0138	0.0340	0.0134
200	0.4983	0.4990	0.4903	0.5050	0.4780
	0.0483	0.0473	0.0552	0.0414	0.0491
	0.0162	0.0167	0.0161	0.0340	0.0143

Table B.13: Results first five SVMs with D2: $m = 100$, $n = 30$, features class 1: $N(10, \sigma)$, features class 2: $N(20, \sigma)$, $0 \leq \sigma \leq 200$, no noise. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

σ	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
0	0.9950	0.9950	0.9950	0.9950	0	0.9950
	0.0051	0.0051	0.0051	0.0051	0	0.0051
	0.0294	0.0139	0.0456	0.0254	0	0.3781
1	0.9947	0.9947	0.9947	0.9947	0	0.9947
	0.0051	0.0051	0.0051	0.0051	0	0.0051
	0.0285	0.0180	0.0189	0.0185	0	0.4039
2	0.9937	0.9937	0.9937	0.9937	0	0.9937
	0.0049	0.0049	0.0049	0.0049	0	0.0049
	0.0305	0.0178	0.0203	0.0176	0	0.3855
5	0.9943	0.9940	0.9940	0.9940	0	0.9940
	0.0057	0.0056	0.0056	0.0056	0	0.0056
	0.0275	0.0216	0.0241	0.0204	0	0.4288
10	0.9747	0.9747	0.9747	0.9747	0	0.9747
	0.0172	0.0172	0.0172	0.0172	0	0.0172
	0.0245	0.0197	0.0249	0.0200	0	0.4200
20	0.8143	0.8137	0.8137	0.8137	0	0.8137
	0.0486	0.0492	0.0492	0.0492	0	0.0492
	0.0373	0.0169	0.0242	0.0209	0	0.3810
50	0	0.6007	0	0	0	0
	0	0.0635	0	0	0	0
	0	0.0126	0	0	0	0
100	0	0.5373	0	0	0	0
	0	0.0540	0	0	0	0
	0	0.0135	0	0	0	0
200	0	0.4930	0	0	0	0
	0	0.0546	0	0	0	0
	0	0.0141	0	0	0	0

Table B.14: Results last six SVMs with D2: $m = 100$, $n = 30$, features class 1: $N(10, \sigma)$, features class 2: $N(20, \sigma)$, $0 \leq \sigma \leq 200$, no noise. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

$noise1, 2$	L_0^a SVM	L_1 SVM	L_2 SVM	Boots. SVM	Cap. L_1 SVM
0	0.9887	0.9743	0.9743	0.9767	0.5653
	0.0111	0.0161	0.0161	0.0217	0.1099
	0.0153	0.0193	0.0176	0.0369	0.0190
0.1	0.9767	0.9353	0.9353	0.9647	0.5573
	0.0154	0.0422	0.0422	0.0354	0.0794
	0.0174	0.0204	0.0193	0.0383	0.0187
0.2	0.9497	0.9113	0.9113	0.9587	0.5617
	0.0451	0.0533	0.0533	0.0410	0.0792
	0.0193	0.0171	0.0212	0.0385	0.0181
0.3	0.9137	0.8613	0.8617	0.9433	0.5567
	0.0691	0.0738	0.0706	0.0551	0.0893
	0.0166	0.0175	0.0187	0.0393	0.0168
0.4	0.8610	0.8373	0.8320	0.9293	0.5313
	0.0881	0.0604	0.0664	0.0523	0.0720
	0.0165	0.0168	0.0181	0.0388	0.0185
0.5	0.6467	0.7583	0.7493	0.8873	0.5453
	0.1559	0.0945	0.1023	0.1010	0.0787
	0.0155	0.0167	0.0174	0.0377	0.0156
0.6	0.5963	0.6987	0.6517	0.8790	0.4960
	0.1343	0.1080	0.1204	0.0911	0.1083
	0.0144	0.0125	0.0155	0.0362	0.0145
0.7	0.5723	0.6467	0.6263	0.8010	0.5547
	0.1292	0.1241	0.1273	0.1386	0.1056
	0.0170	0.0152	0.0182	0.0354	0.0142
0.8	0.4913	0.5100	0.4840	0.6637	0.5003
	0.1001	0.1026	0.1206	0.1847	0.0909
	0.0174	0.0171	0.0176	0.0359	0.0159
0.9	0.4890	0.4973	0.5017	0.5630	0.5050
	0.0882	0.0902	0.0973	0.1614	0.0847
	0.0176	0.0137	0.0190	0.0372	0.0165
1	0.4940	0.5013	0.5077	0.5207	0.4920
	0.0883	0.0940	0.0811	0.0966	0.0752
	0.0183	0.0175	0.0194	0.0353	0.0168

Table B.15: Results first five SVMs with D2: $m = 100$, $n = 30$, features class 1: $N(10, 20)$, features class 2: $N(20, 20)$, noise: $N(0, 100)$, $0 \leq noise1 = noise2 \leq 1$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.

$noise1, 2$	L_0^b SVM	L_2L_1 SVM	$L_2L_0^b$ SVM	$L_1L_0^b$ SVM	SCAD SVM	El. SCAD SVM
0	0.9743	0.9743	0.9743	0.9743	0	0.9743
	0.0161	0.0161	0.0161	0.0161	0	0.0161
	0.0235	0.0175	0.0212	0.0186	0	0.3365
0.1	0.9350	0.9353	0.9353	0.9353	0	0.9353
	0.0420	0.0422	0.0422	0.0422	0	0.0422
	0.0294	0.0184	0.0230	0.0188	0	0.3691
0.2	0.9117	0.9113	0.9113	0.9113	0	0.9113
	0.0532	0.0533	0.0533	0.0533	0	0.0533
	0.0407	0.0180	0.0250	0.0222	0	0.3856
0.3	0.8697	0.8620	0.8617	0.8583	0	0
	0.0673	0.0703	0.0706	0.0756	0	0
	0.1006	0.0142	0.0462	0.0343	0	0
0.4	0	0.8293	0.8320	0.8403	0	0
	0	0.0685	0.0664	0.0584	0	0
	0	0.0133	0.1086	0.0994	0	0
0.5	0	0.7473	0.7477	0.7553	0	0
	0	0.1073	0.1017	0.0961	0	0
	0	0.0144	0.2207	0.1931	0	0
0.6	0	0.6510	0.6510	0	0	0
	0	0.1234	0.1204	0	0	0
	0	0.0108	0.3731	0	0	0
0.7	0	0.6250	0	0	0	0
	0	0.1273	0	0	0	0
	0	0.0129	0	0	0	0
0.8	0	0.4847	0	0	0	0
	0	0.1163	0	0	0	0
	0	0.0126	0	0	0	0
0.9	0	0.5007	0	0	0	0
	0	0.0986	0	0	0	0
	0	0.0117	0	0	0	0
1	0	0.5070	0	0	0	0
	0	0.0791	0	0	0	0
	0	0.0136	0	0	0	0

Table B.16: Results last six SVMs with D2: $m = 100$, $n = 30$, features class 1: $N(10, 20)$, features class 2: $N(20, 20)$, noise: $N(0, 100)$, $0 \leq noise1 = noise2 \leq 1$. The first, second and third rows are the score, std. dev. and runtime respectively. A result 0 indicates that the model took longer than a second to run.