Programa de doctorado "Matemáticas"

PhD Dissertation

NEW ADVANCES IN DATA SCIENCE PROBLEMS THROUGH HYPERPLANES LOCATION

Author

Alberto Japón Sáez

Supervisors

Prof. Dr. *Justo Puerto Albandoz*

Prof. Dr. *Víctor Blanco Izquierdo*

March 1, 2022

Resumen

El trabajo de esta tesis se centra en desarrollar nuevas metodologías para abordar problemas clásicos de Ciencia de Datos desde el punto de vista de la Teoría de la Localización. En particular, nos centramos en problemas de localización de hiperplanos que obtenemos modelando y resolviendo problemas de programación lineal, y no lineal, entera mixta.

El Capítulo 1 presenta las técnicas desarrolladas en la literatura desde las que iniciamos este trabajo, que comprenden Máquinas de Vector Soporte, Árboles de Clasificación y Teoría de Ajuste de Hiperplanos.

El Capítulo 2, basado en el artículo Blanco et al. (2020b), se centra en el estudio de localizar un conjunto de hiperplanos para resolver un problema de clasificación multiclase, extendiendo las Máquinas de Vector Soporte al escenario multiclase. Presentamos cuatro formulaciones resultado de combinar dos formas de medir los errores de clasificación y las normas utilizadas para medir las distancias. Reportamos los resultados de experimentos en conjuntos de datos reales y sintéticos donde se muestra la alta capacidad predictiva de nuestros modelos frente a otros existentes en la literatura. Además, demostramos también que el *truco del kernel* es aplicable en nuestro método.

En el Capítulo 3, inspirado en el artículo Blanco et al. (2021a), también tratamos el problema de localizar un conjunto de hiperplanos, sin embargo, en este caso lo hacemos bajo la premisa de minimizar una función objetivo basada en el ajuste de distancias del conjunto de puntos a los hiperplanos. El problema lo tratamos en un contexto general donde las distancias de los puntos a los hiperplanos se contabilizan mediante funciones de tipo mediana ordenada (Ordered Weighted Averaging-OWA). Para la resolución del problema reportamos una formulación compacta y otra basada en conjuntos de las observaciones, para la cual desarrollamos un método de generación de columnas. Reportamos numerosos experimentos que permiten evaluar las dos metodologías presentadas. Discutimos resultados teóricos sobre la escalabilidad de los problemas y reportamos también algunos análisis geométricos de las soluciones.

El Capítulo 4, que sigue el trabajo planteado en Blanco et al. (2020a), trata el problema de encontrar un hiperplano separador de margen máximo con la particularidad de que consideramos que puede haber cierto ruido en las etiquetas de la muestra de entrenamiento. Desarrollamos tres metodologías, dos de ellas basadas en técnicas de cluster, que incorporan en el entrenamiento la capacidad de reetiquetar observaciones, es decir, de considerar observaciones como pertenecientes a su clase contraria. En los resultados computacionales mostramos como nuestros modelos obtienen mayores tasas de acierto en la clasificación cuando hay ruido aleatorio en las etiquetas de las muestras de entrenamiento.

Los capítulos 5 y 6, basados en los trabajos presentados en Blanco et al. (2021c) y Blanco et al. (2021b) respectivamente, se centran en el estudio de Árboles de Clasificación en la que las ramas se definen mediante hiperplanos basados en Máquinas de Véctor Soporte. Las metodoloías desarrolladas en estos capítulos heredan propiedades de los métodos presentados en el Capítulo 4, que forman un pilar esencial en los mismos. Por un lado, en el Capítulo 5 se centra en el estudio de problemas de clasificación binaria donde además se contempla la posibilidad de existencia de ruido aleatorio en las etiquetas de las observaciones en las muestras de entrenamiento. Por otro lado, el Capítulo 6 se centra en el estudio de problemas de clasificación multiclase. En ambos capítulos se presentan experimentos computacionales que muestran mejoras en las predicciones sobre bases de datos reales frente a otros métodos de la literatura.

Finalmente, en el Capítulo 7 se muestran las conclusiones de la tesis.

Abstract

This thesis dissertation focus on developing new approaches for different Data Science problems from a Location Theory perspective. In particular, we concentrate on locating hyperplanes by means of solving Mixed Integer Linear and Non Linear Problems.

Chapter 1 introduces the baseline techniques involved in this work, which encompass Support Vector Machines, Decision Trees and Fitting Hyperplanes Theory.

In Chapter 2, which is based in Blanco et al. (2020b), we study the problem of locating a set of hyperplanes for multiclass classification problems, extending the binary Support Vector Machines paradigm. We present four Mathematical Programming formulations which allow us to vary the error measures involved in the problems as well as the norms used to measure distances. We report an extensive battery of computational experiment over real and synthetic datasets which reveal the powerfulness of our approach. Moreover, we prove that the *kernel trick* can be applicable in our method.

Chapter 3, which is inspired in Blanco et al. (2021a), also focus on locating a set of hyperplanes, in this case, aiming to minimize an objective function of the closest distances from a set of points. The problem is treated in a general framework in which norm-based distances between points and hyperplanes are aggregated by means of ordered median functions. We present a compact formulation and also a set partitioning one. A column generation procedure is developed in order to solve the set partitioning problem. We report the results of an extensive computational experience, as well as theoretical results over the scalability issues and geometrical analysis of the optimal solutions.

Chapter 4, which follows the work presented in Blanco et al. (2020a), addresses the problem of finding a separating hyperplane for binary classification problems in which label noise is considered to occur over the training sample. We derive three methodologies, two of them based on clustering techniques, which incorporate the ability of relabeling observations, i.e., treating them as if they belong to their contrary class, during the training process. We report computational experiments that show how our methodologies obtain higher accuracies when training samples

contain label noise.

Chapters 5 and 6, which are based in the works Blanco et al. (2021c) and Blanco et al. (2021b), respectively, consider the problem of locating a set of hyperplanes, following the Support Vector Machines classification principles, in the context of Classification Trees. The methodologies developed in both chapters inherit properties from Chapter 4, which play an important role in the problems formulations. On the one hand, Chapter 5 focuses on binary classification problems where label noise can occur in training samples. On the other hand, Chapter 6 focus on solving the multiclass classification problem. Both chapters present the results of our computational experiments which show how the methodologies derived outperform other Classification Trees methodologies.

Finally Chapter 7 presents the conclusions of this thesis.

Contents

Chapter 1

Introduction

Historically, Mathematics has been considered an abstract field which has no, or at best little, relation to the sensitive world for a majority of society. This is no longer true. The collective conscience about the generation and use of data has implied a change in the perception of mathematical knowledge amongst people. Nowadays, words like algorithms, Machine Learning, Artificial Inteligence or Big Data are not unlikely to appear in a casual conversation. At the same time, proper data management has proven to be an essential requirement for decision making, and according to this, institutions spend a large amount of resources on such a purpose.

Statistics and Operations Research provide an extensive pool of techniques so as to deal with data. When facing a data problem in real life, two scenarios are mainly presented: On the one hand there are cases where, guaranteeing to provide a good solution, computational time is desired to be as small as possible. On the other hand, there are problems in which the quality of the solution is the most important part, therefore the optimal solution must be found despite the fact that it can be computationally expensive. This optimal solution is usually obtained by means of solving an optimization problem. In this thesis we focus on deriving and solving optimization problems in which a set of hyperplanes are located in order to solve some Data Science problems.

Although Data Science does not have a concrete definition, authors usually rely on the same ideas. Before Data Science was even named, Tukey stated (Tukey, 1962):

> *By and large, the great innovations in statistics have not had correspondingly great effects upon data analysis. The extensive calculation of "sum of squares" is the outstanding exception. [...] Is it not time to seek out novelty in data analysis?*

Tukey pointed out the need of a new way of analysing data without taking into account as much statistical constraints as usually done, encouraging scientists to draw hypotheses directly from data and not necessarily to reduce the problem to finding a statistical model fitting the data. Data Science is later on introduced, in the late 1990s, as a discipline encompassing the path from data colection to decision making passing through data modeling, combining Statistics and Computer Science with the main objective of helping in the decision making process.

Data science is currently experiencing an outstanding time. Technology is allowing computers to go further every day. Decision making is becoming a data driven procces throughout industries and organizations around the whole world. Data itself is often a valuable asset ahead of other tangible assets. Accordingly, Data Science has attracted the interest of people and a lot of new techniques are constantly being developed.

Classification and regression problems represent some of the most challenging problems within Data Science. In classification problems we consider given a training sample in the form $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times \{1, \ldots, k\}$, where a set of p features has been measured for a set of n individuals $(x_1, \ldots, x_n)$, as well as a label $(y_1, \ldots, y_n)$ identified with an element of a finite set of k classes. The goal of supervised classification is to derive a decision rule $D_{\mathcal{X}} : \mathbb{R}^p \to \{1, \ldots, k\}$ in order to accurately predict the labels of the forthcoming unobserved data. We call it a multiclass classification problem when the number of classes is greater than two, and we call it a binary classification problem when only two classes are involved. When facing a binary classification problem, the notation on the labels set $\{1, 2\}$ is often replace by $\{-1, +1\}$, and hence the classes are refered to as the negative and the positive ones. On its continuous counterpart, in regression problems the label is replaced by a continuous target variable. Therefore, training samples are given in the form $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times \mathbb{R}$, where the decision rule $D_{\mathcal{X}} : \mathbb{R}^p \to \mathbb{R}$ returns a real number as a prediction. Many different classification and regression techniques can be found in the literature, as for instance, Logistic Regression and Linear Discriminant Analysis (Friedman (2017)), Classification and Regression Trees (Breiman et al. (2017)), Support Vector Machines (Cortes and Vapnik (1995)) and Support Vector Regression (Drucker et al. (1997)), k-nearest neighbor (Cover and Hart (1967)), Random Forest (Breiman (2001)) or Deep Learning methods (Goodfellow et al. (2016)). In addition to supervised problems, there are Data Science problems which lie in the perspective of an unsupervised paradigm, i.e., problems where no target variables are presented. Cluster problems are the most famous ones. The goal on these problems is to create a finite number of groups which present, in a certain sense, homogeonity within the elements of the groups and at the same time heterogenity when elements from different groups are compared. In such a context, the k-means algorithm or hierarchical clusters are the most popular baseline techniques, nevertheless, more complex methods have been studied in the literature as for instance the ordered p-median problem (Boland et al. (2006); Kalcsics et al. (2002); Labbé et al. (2017)).

Throughout the chapters of this thesis the focus is set on developing new strategies for classification and regression problems. However, cluster methodologies are not left out and they are present in most of the chapters. The main elements from where we start this work involve Support Vector Machines, Decision Trees and Fitting Hyperplanes Theory. The following sections of this introduction are devoted to introduce these elements as well as to detail the contributions of this dissertation.

1.1 Support Vector Machines

1.1.1 Original problem

Support Vector Machines (SVM), introduced by Cortes and Vapnik (1995), is a technique originally proposed to approach binary classification problems by means of solving an optimization problem. Given a training sample $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times \{-1, +1\}$, the goal of SVM is to obtain a hyperplane separating the data ($x \in \mathbb{R}^p$) into their two different classes ($y \in \{-1, +1\}$). Among all possible hyperplanes that can obtain such a separation between the classes, SVM looks for the one with maximum margin (maximum distance from classes to the separating hyperplane) while minimizing the misclassification errors. Let us denote by $\mathcal{H}$ a hyperplane in $\mathbb{R}^p$ in the form $\mathcal{H} = \{z \in \mathbb{R}^p \ : \ \omega' z + \omega_0 = 0\}$ for some $\omega \in \mathbb{R}^p$ and $\omega_0 \in \mathbb{R}$ (the vector v' is the result of the transpose operator applied to the vector $v \in \mathbb{R}^p$). This hyperplane will induce a subdivision of the data space $\mathbb{R}^p$ into three regions: the $+1$ (positive) half-space $\mathcal{H}^+ = \{z \ : \ \omega' z + \omega_0 > 1\}$, the -1 (negative) half-space $\mathcal{H}^- = \{z \ : \ \omega' z + \omega_0 < -1\}$ and the strip $\mathcal{S} = \{z \ : -1 \leq \ \omega' z + \omega_0 \leq 1\}$. In the SVM model, positive-class observations ($y = +1$) will be forced to lie on the positive half-space, and the same constraint will be imposed for the negative-class ($y = -1$) observations on the negative half-space. Moreover, observations will also be penalized if they lie inside the strip. When these constraints are violated for an observation, a classification error is accounted for in the optimization problem. The separation (margin) between classes is computed as the width of the strip $\mathcal{S}$. The points defining the width of the strip, which are known as the support vectors, verify $|\omega' x_i + \omega_0| = 1$. Therefore, recalling that given a point $z \in \mathbb{R}^p$, its Euclidean distance to the hyperplane $\mathcal{H}$ is $\dfrac{|\omega' z + \omega_0|}{\|\omega\|_2}$, we obtain that the width of the strip is given by $\dfrac{2}{\|\omega\|_2}$. As mentioned before, the SVM separating hyperplane will be obtained from an equilibrium of maximizing the separation between classes and minimizing these classification errors. Denoting by $d_i \in \mathbb{R}^+$ the classification error of observation i, and by c the penalty for these errors, the SVM can be formulated as the following Non Linear Problem (NLP):

$$
\begin{aligned}
\min \ & \frac{1}{2}\|\omega\|_2^2 + c\sum_{i=1}^{n} d_i && && \text{(SVM)}\\
\text{s.t.} \ & y_i(\omega' x_i + \omega_0) \geq 1 - d_i, && \forall i = 1, \ldots, n,\\
& d_i \geq 0, && \forall i = 1, \ldots, n,\\
& \omega \in \mathbb{R}^p, \ \omega_0 \in \mathbb{R}.
\end{aligned}
$$

In agreement with this, the decision rule for out-of-sample observations, $D_{\mathcal{X}}$, arises naturally according to the region in which an observation is located:

$$D_{\mathcal{X}}(z) = \begin{cases} -1 & \text{if } \omega' z + \omega_0 < 0, \\ 1 & \text{if } \omega' z + \omega_0 > 0. \end{cases}$$

We see in Figure 4.1 a graphical example of the elements involved in a SVM problem. Classes are being represented by the blue and green colors. The black line is the separating hyperplane, and the dashed parallels are the boundary of the strip. The two blue points and the two green points which lie on these lines are the support vectors of this solution. Moreover, the red lines are being used to remark the two kind of errors that can be accounted for in the problem. On the one hand, the small red line is representing a penalization of a blue observation because, despite the fact that it is being well classified, it lies inside the margin, which is desired to be an isolated area. On the other hand, the big red line is representing a penalization of a blue observation which is being wrong classified, since it lies in the side of the green observations.

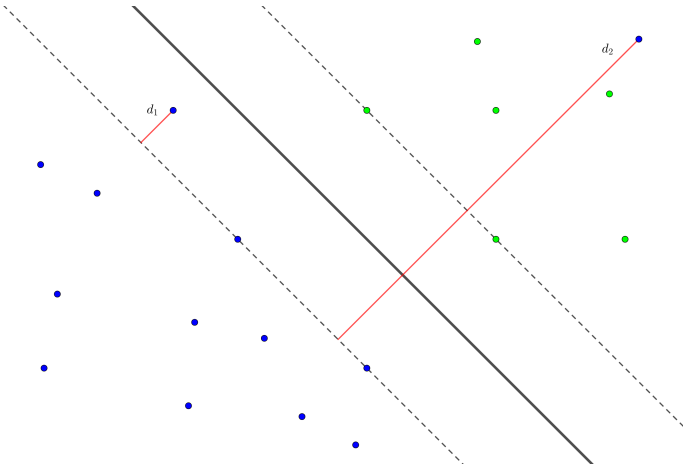


Figure 1.1: SVM example

SVM is one of the most famous classification tools. Nevertheless, this is not only due to its original (linear) formulation, but also because of the so called kernel trick. When facing a problem which is not linearly separable, i.e., it is imposible to obtain a hyperplane separating the classes with no classification errors, or if the amount of classification errors obtained when using a separating hyperplane is too big, SVM provides a very useful (kernel) trick in order to deal with these situations. This trick relies on the idea of proyecting data onto a higher dimensional space in which data can be linearly separable, or where at least misclassification errors are not as big as in the original space. This higher dimensional space is called the feature space. In Figure 1.2 we see an example of some points that can not

be linearly separated in their original data space (left), meanwhile a hyperplane is separating them in the feature space (right). The advantages of this kernel trick

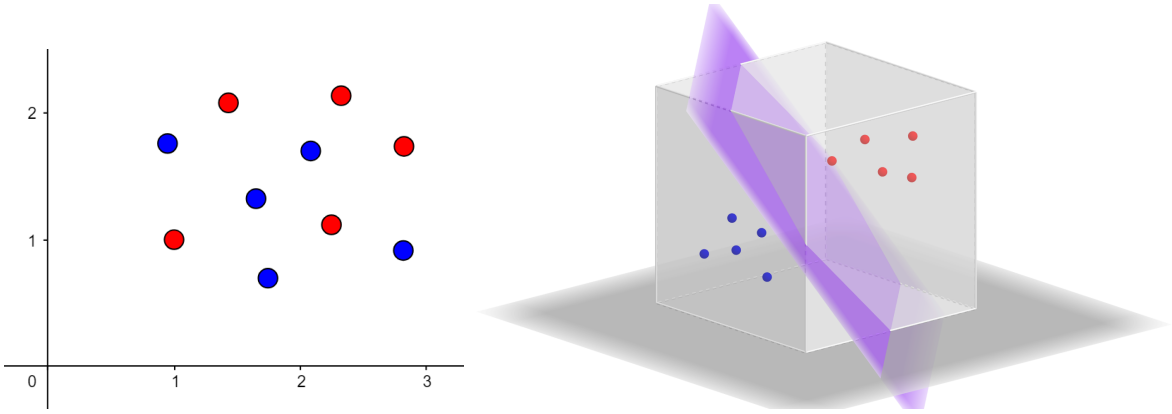


Figure 1.2: Original space (left) and feature space with a hyperplane (right)

are that, for a transformation $\varphi : X \to \mathcal{F}$ from the original space X to the feature space $\mathcal{F}$, there is no need of knowing the exact transformation φ, it is enough with the knowledge of the form of the inner products on the feature space. Moreover, the computational cost of solving the problem in $\mathcal{F}$ is similar to the cost of solving the original problem. These statements can be justified by formulating the dual (Lagrangean) of the problem. In order to obtain this dual problem, we must start from its primal, which can be written as follows,

$$\min \frac{\omega'\omega}{2} + c\sum_{i=1}^{n} d_i \qquad (\mathcal{F}-\text{SVM})$$

$$\begin{aligned} \text{s.t.} \quad & y_i(\omega'\varphi(x_i) + \omega_0) \geq 1 - d_i, && \forall i = 1, \ldots, n, \\ & d_i \geq 0, && \forall i = 1, \ldots, n, \\ & \omega \in \mathcal{F}, \ \omega_0 \in \mathbb{R}, \end{aligned}$$

and then to apply the Karush-Kuhn-Tucker (KKT) conditions. Therefore, if we consider the following set of dual variables $\alpha = (\alpha_1, \ldots, \alpha_n) \geq 0$, $\lambda = (\lambda_1, \ldots, \lambda_n) \geq 0$, and the Lagrangean function $\mathcal{L}(\omega, \omega_0, \alpha, \lambda, d)$ defined as

$$\mathcal{L}(\omega, \omega_0, \alpha, \lambda, d) = \frac{\omega'\omega}{2} + c\sum_{i=1}^{n} d_i - \sum_{i=1}^{n} \alpha_i \left[y_i(\omega'\varphi(x_i) + \omega_0) - (1 - d_i) \right] - \sum_{i=1}^{n} \lambda_i d_i,$$

where $d = (d_1, \ldots, d_n)$ is the vector of the error variables, we obtain by differentiating with respect to ω, ω_0 and d_i, and equalazing to zero, the following optimality

conditions:

$$\frac{\partial \mathcal{L}(\omega, \omega_0, \alpha, \lambda, d)}{\partial \omega} = 0 \implies \omega = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i),$$
$$\frac{\partial \mathcal{L}(\omega, \omega_0, \alpha, \lambda, d)}{\partial \omega_0} = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0,$$
$$\frac{\partial \mathcal{L}(\omega, \omega_0, \alpha, \lambda, d)}{\partial d_i} = 0 \implies \alpha_i = c - \lambda_i.$$

According to this, substituting these conditions on $\mathcal{L}(\omega, \omega_0, \alpha, \lambda, d)$, the dual problem is derived as follows:

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \varphi(x_i)' \varphi(x_j)$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & \alpha_i = c - \lambda_i, && \forall i = 1, \ldots, n, \\ & 0 \leq \lambda_i, && \forall i = 1, \ldots, n, \\ & 0 \leq \alpha_i, && \forall i = 1, \ldots, n, \end{aligned}$$

which can be rewritten as

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \varphi(x_i)' \varphi(x_j) \qquad (\text{Dual} - \mathcal{F} - \text{SVM})$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq c, && \forall i = 1, \ldots, n. \end{aligned}$$

Hence, as we can see, data dependency is shown only through the inner products in $\mathcal{F}$. Therefore, the kernel trick is based on using a known non linear function,

$$K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$$
$$(x_i, x_j) \to \varphi(x_i)' \varphi(x_j),$$

instead of computing the inner products of the transformed data, mantaining this way the computational cost of solving the original problem and avoiding us to know

the explicit form of φ.

Finally, we conclude this SVM presentation with a brief discussion about different error measures that can be used when solving the problem. The formulations presented before attend to an unbounded continuous meassure of the errors through the d_i variables, which are referred to as Hinge Loss (HL) measures. SVM can struggle dealing with outlier observations when minimizing HL error measures. This is due to the fact that, in extreme cases, the sum of loads of little classification errors can be smaller than the sum of a few big errors. With the aim of obtaining robust SVM models against outlier observations, Brooks introduced in (Brooks (2011)) two approaches which provide an upper bounded error meassure by means of solving a Mixed Integer Non Linear Programming Problem (MINLP). The first approach is the Ramp Loss SVM (RL-SVM), which maintains the same value as standard SVM on the d_i variables for observations within the margin, but considers these errors to be constant for wrong classified observations outside the margin. In order to model the problem, we need to introduce the following set of binary variables $\xi_i \in \{0, 1\}$ for each of the observations defined as:

$$\xi_i = \begin{cases} 1 & \text{if } x_i \text{ is wrong classified outside the margin,} \\ \\ 0 & \text{otherwise.} \end{cases}$$

According to this, the RL-SVM is formulated as follows:

$$\min \frac{1}{2}\|\omega\|_2^2 + c\left(\sum_{i=1}^{n} d_i + 2\sum_{i=1}^{n} \xi_i\right) \qquad (\text{RL} - \text{SVM})$$

$$\begin{aligned} \text{s.t.} \quad & y_i(\omega' x_i + \omega_0) \geq 1 - d_i - M\xi_i, && \forall i = 1, \ldots, n, \\ & 0 \leq d_i \leq 2, && \forall i = 1, \ldots, n, \\ & \xi_i \in \{0, 1\}, && \forall i = 1, \ldots, n, \\ & \omega \in \mathbb{R}^p,\ \omega_0 \in \mathbb{R}, \end{aligned}$$

where M is a big enough constant (see Baldomero-Naranjo et al. (2020) for further details on estimating this constant), and c is the penalization cost. The second approach is the Hard-Loss SVM, which is rather similar to the RL-SVM with the difference that in this method, all classification errors are considered to be constant regardless of the distance from the observations to the separating hyperplane.

Therefore, in this problem ξ_i variables are slightly modified and defined as:

$$\xi_i = \begin{cases} 1 & \text{if } x_i \text{ is wrong classified or is located inside the margin,} \\ \\ 0 & \text{otherwise,} \end{cases}$$

and accordingly, being M a big enough constant and c the penalization cost, the problem is formulated as follows:

$$\min \frac{1}{2}\|\omega\|_2^2 + c\sum_{i=1}^{n}\xi_i \qquad (\text{HardLoss} - \text{SVM})$$

$$\begin{aligned} \text{s.t.} \quad & y_i(\omega' x_i + \omega_0) \geq 1 - M\xi_i, && \forall i = 1, \ldots, n, \\ & \xi_i \in \{0, 1\}, && \forall i = 1, \ldots, n, \\ & \omega \in \mathbb{R}^p, \ \omega_0 \in \mathbb{R}. \end{aligned}$$

1.1.2 Multiclass approaches

The good perfomance of SVM in binary classification problems has motivated the study of its extension to the multiclass scenario from different points of view. There is no a unique extension to this problem because when dealing with more than two classes, some elements which were unequivocally defined for the biclass problem are left open to interpretation. When facing a multiclass problem, as for instance the one shown in Figure 1.3 of a three class problem in $\mathbb{R}^2$, some questions come up naturally: How many hyperplanes do we need to solve the problem? What is the margin in this context? How do we define classification errors? How do we define a decision rule?

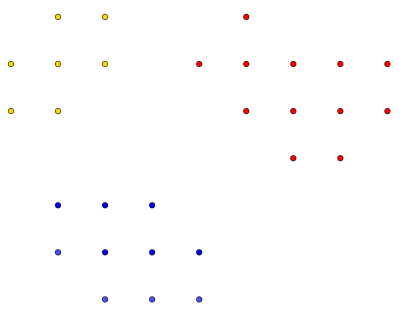


Figure 1.3: Multiclass example

There are two main lines to approach the problem. On the one hand, there are sequential methods in which the multiclass problem is divided into a set of binary ones, which are afterwards combined in order to create a solution for the original problem. On the other hand, there are global methods which provide a solution by solving a single optimization problem. In the following paragraphs we describe some of the most known multiclass SVM methods.

• **Sequential methods**

The first method we introduce in this section is the One Versus All (OVA) SVM. In such a method, considering k possible classes, the problem is divided into k binary classification subproblems. Each of these subproblems consists on separating one class from all the other classes. Therefore, positive class is defined as the class to be separated, and the negative one is a fictitious class which is formed gathering the remaining classes. For the multiclass example presented in Figure 1.3, we show in Figure 1.4 the first hyperplane of the OVA method, resulting from separating the yellow class from the other two classes (represented together by the black color).

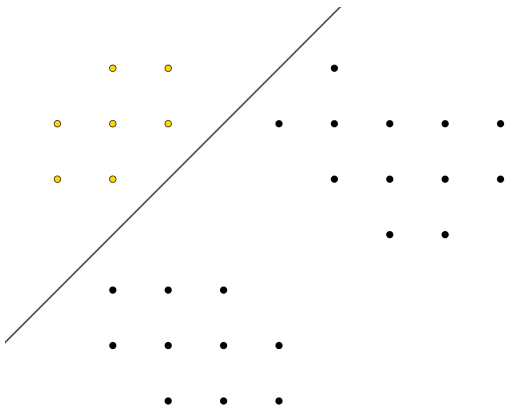


Figure 1.4: OVA - yellow class versus rest

This process is repeated until all the hyperplanes $\mathcal{H}_t = \{z \in \mathbb{R}^p \ : \ \omega_t' z + \omega_{t0} = 0\}$, $\omega_t \in \mathbb{R}^p$, $\omega_{t0} \in \mathbb{R}$, $t = 1, \ldots, k$, are located, and the decision rule is derived as follows:

$$D_{\mathcal{X}} \to \{1, \ldots, k\}$$
$$x \to \arg \max_{t=1,\ldots,k} \left(\omega_t' x + \omega_{t0}\right)$$

The remaining subproblems are shown in Figure 1.5, and the three hyperplanes combined together in Figure 1.6.

The main advantage of OVA is that it is a fast method to calculate due to the fact that only k binary problems are involved. Nevertheless, this method can struggle when the number of classes is large. For a large number of classes, the resulting

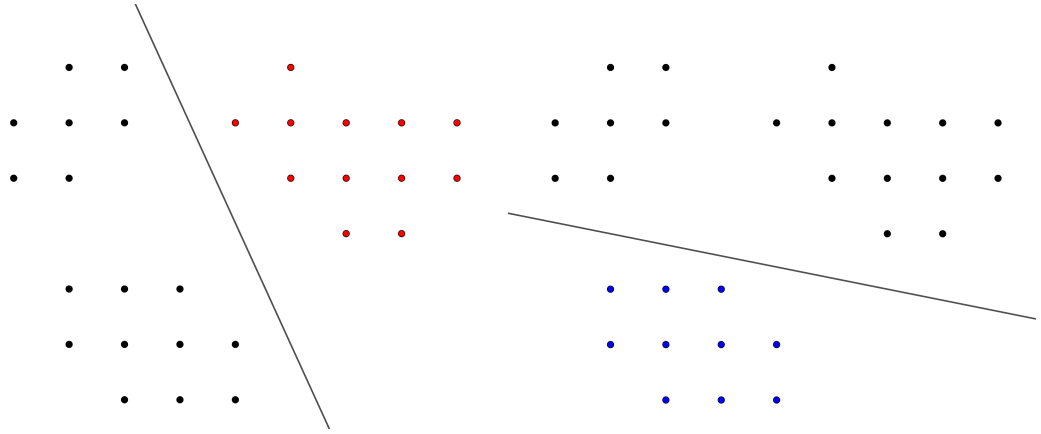


Figure 1.5: OVA - red class versus rest (left) and blue class versus rest (right)

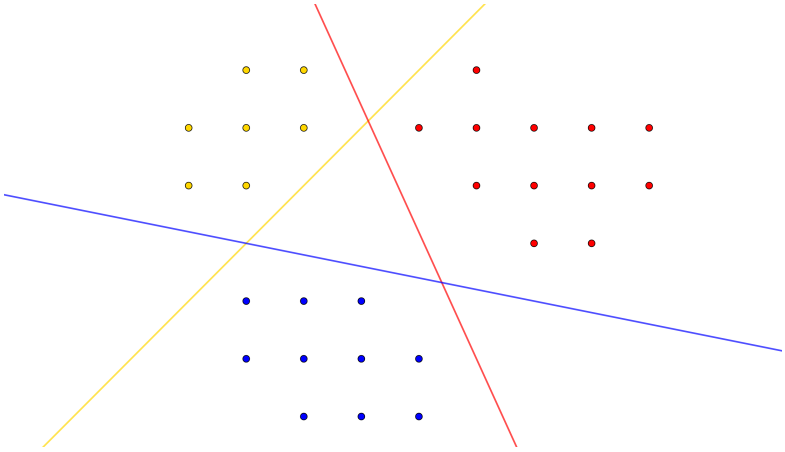


Figure 1.6: OVA - final solution

binary subproblems can be very unbalanced, and the combined result of solving these unbalanced problems can lead to bad global results. The second method we introduce in this section avoids these unbalanced problems by separating classes one to one. This method is known as One Versus One (OVO) SVM. Therefore, in a k classification problem, the number of binary subproblems to solve increases to $\binom{k}{2}$. Returning to the example presented in Figure 1.3, we see in Figure 1.7 the subproblem derived from separating the yellow and the red classes. Furthermore, we see in Figure 1.8 all the hyperplanes involved in the OVO solution. Regarding to the decision rule, we find a big difference with respect to the OVA method. In OVO, for a given observation to be predicted, we calculate all the predictions obtained by each of the binary subproblems. The final prediction of the method is computed afterwards as the most represented prediction amongst the binary ones.

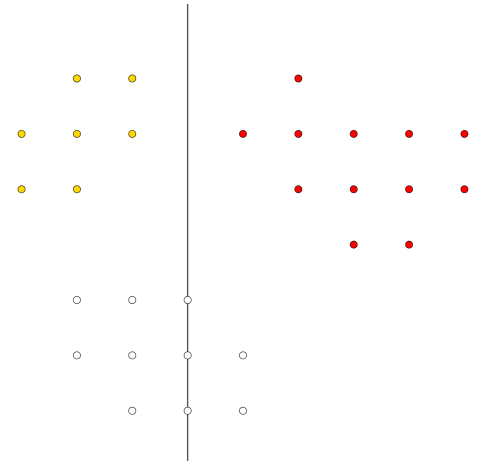


Figure 1.7: OVO - yellow class versus red class

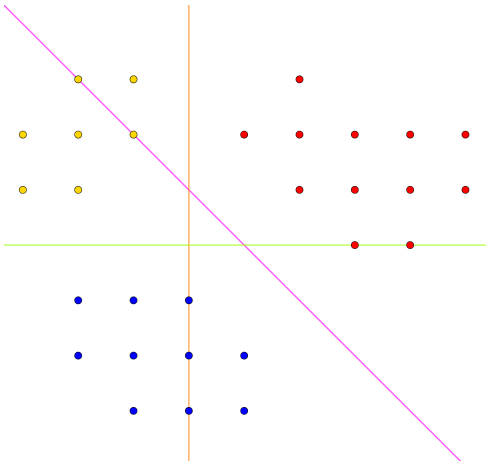


Figure 1.8: OVO - final solution

- **Global methods**

There are methods which obtain the multiclass SVM solution by solving a single optimization problem. In this section we present two of the most famous ones. Both of these methods rely on the same idea of OVA SVM, i.e., separating one class from the rest. Nevertheless, these methods do not have the problems OVA had with unbalanced subproblems since all the classes and all the classification errors are involved at the same time in the global optimization. The first of the methods was proposed by Weston and Watkins (1998). In this method there are $k-1$ error variables per observation, $d_{it},\ i = 1,\ldots,n,\ t = 1,\ldots,k-1$, and k hyperplanes

involved. Weston and Watkins (WW) formulation is derived as follows:

$$\begin{aligned} & \min \frac{1}{2}\sum_{t=1}^{k}\|\omega_t\|_2^2 + c\sum_{i=1}^{n}\sum_{t\neq y_i} d_{it} && && \text{(WW)} \\ \text{s.t. } & \omega'_{y_i}x_i + \omega_{y_i 0} \geq \omega'_t x_i + \omega_{t0} + 2 - d_{it}, && \forall i = 1,\dots,n,\ t \in \{1,\dots,k\}\diagdown y_i, \\ & d_{ij} \geq 0, && \forall i = 1,\dots,n,\ t \in \{1,\dots,k\}\diagdown y_i, \\ & \omega_t \in \mathbb{R}^p,\ \omega_{t0} \in \mathbb{R}, && \forall t = 1,\dots,k, \end{aligned}$$

where c is the penalization cost. Despite the fact that this method has proven to obtain good results for many multiclass problems, managing more than one error variable per observation can lead to some troubles when dealing with outlier observations. A few years later, Crammer and Singer proposed in (Crammer and Singer (2001)) a similar method which only requires an error variable per observation. Nevertheless, in order to reduce the number of error variables, some binary variables need to be added to the problem. Such a set of variables is defined follows:

$$\delta_{y_i t} = \begin{cases} 1 & \text{if } y_i \neq t, \\ \\ 0 & \text{if } y_i = t. \end{cases}$$

Therefore, Crammer and Singer (CS) model is formulated as follows:

$$\begin{aligned} & \min \frac{1}{2}\sum_{t=1}^{k}\|\omega_t\|_2^2 + c\sum_{i=1}^{n} d_i && && \text{(CS)} \\ \text{s.t. } & \omega'_{y_i}x_i + \delta_{y_i t} - \omega'_t x_i \geq 1 - d_i, && \forall i = 1,\dots,n,\ t = 1,\dots,k, \\ & d_i \geq 0, && \forall i = 1,\dots,n, \\ & \omega_t \in \mathbb{R}^p,\ \omega_{t0} \in \mathbb{R}, && \forall t = 1,\dots,k. \end{aligned}$$

WW and CS approaches have some similarities, both problems are popular because of their good performance, the solutions are relatively fast to calculate, both problems share the same decision rule as the OVA method, and both methods seem to have missed the same little margin optimization detail: maximizing the margin amongst the k hyperplanes can be computed as $\max \sum_{t=1}^{k} \frac{2}{\|\omega_t\|_2^2}$, and that maximization does not always equate to compute $\min \frac{1}{2}\sum_{t=1}^{k}\|\omega_t\|_2^2$, which is the term appearing in both objective functions. This detail is further discussed in Chapter 2.

1.2 Classification Trees

Classification Trees (CT) are a family of classification methods based on a hierarchical relation among a set of nodes. CT involve a set of branches conecting the nodes and defining the paths that observations can take by following a tree graph scheme. At the first stage of a CT method all the observations belong to one node, which is known as the root node. From this node, branches are sequentially created by splits on the feature space, creating intermediate nodes (which are called branch nodes) until the terminal nodes (which are called leaf nodes) are reached. The predicted label for an observation is given by the majority class of the leaf node where it is located. Observations fall from the root node to the leaf nodes visiting a certain number of nodes. The maximum number of nodes an observation can visit within the tree is referred to as the tree depth.

Specifically, at each branch node, t, of the tree a hyperplane $\mathcal{H}_t = \{z \in \mathbb{R}^p : \omega_t' z + \omega_{t0} = 0\}$ is constructed and the feature space splits are defined as $\omega_t' z + \omega_{t0} < 0$ (left branch) and $\omega_t' z + \omega_{t0} \geq 0$ (right branch). In Fig. 1.9 we show a simple classification tree with depth two, for a small dataset with 6 observations, all of them correctly classified in the leaf nodes.

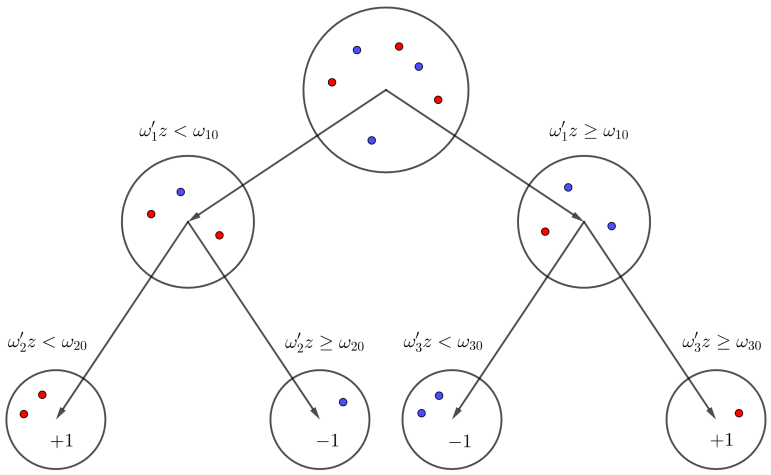


Figure 1.9: Decision tree of depth two.

On the one hand, there are many different heuristic approaches to build a CT. The most popular method is CART, introduced by Breiman et al. (1984), which is to be described in next section. However, we can also find in the literature other well known algorithms as for instance $C4.5$ (Salzberg (1994)) or $ID3$ (Quinlan (1996)). On the other hand, Bertsimas and Dunn (2017) have recently proposed an optimal approach to build a CT by solving a Mathematical Programming problem. This classification method is further detailed in 1.2.2.

1.2.1 CART

CART is a greedy heuristic approach which myopically constructs the CT without further foreseen to deeper nodes. Starting at the root node, it constructs the splits by means of hyperplanes minimizing an impurity measure at each of the branch nodes. Each split results in two new nodes, and this procedure is repeated until a stopping criteria is reached. These criteria normally encompass some of the following:

- Reaching the maximum allowed depth in the tree.
- Obtaining a node with only one class represented amongst its observations.
- Obtaining less than a minimun number of observations required to be in a node.
- Imposibility to obtain a reduction of the impurity measure.

Gini index (Gini (1912)) is used in CART as impurity measure. In a CART classification problem with k classes, denoting by p_{tq} the probability of belonging to class q in node t, the Gini index is obtained by the following sum:

$$\sum_{q=1}^{k} p_{tq}(1 - p_{tq}).$$

Therefore, a new split is performed in node t if the above sum is greater than the average of the analogous sums over its resulting child nodes. According to this top-down procedure, CART may lead to deep trees if a maximum depth is not stablished. The problem with very deep trees is that these may lead to making mistakes on predictions in out-of-sample observations because of overfitting over the training sample. Hence, trees are normally subject to a prune process based on the trade-off between the impurity function reduction and a cost-complexity parameter.

The main advantage of CART is that it is easy to implement and it is also fast to train. Moreover, the solution is often constrained to univariate splits, i.e., splits in which only one of the hyperplane coefficients is allowed to be different from zero. These univariate splits are used to ease interpretability in the model. In practise, it is easier to understand whether a variable is greater or lower than a certain number rather than analyzing an oblique hyperplane (i.e., a hyperplane with freedom in its coefficients).

Nevertheless, the main disadvantage of CART is that, based on its top-down greedy nature, some good splits according to a global classification in the leaf nodes might never be found because of some good previous local splits. In fact, this weakness has motivated different heuristic approaches to rank the variable importance amongst a dataset. These are usually based on using different subsamples (varying

on the number of observations or variables) to build the trees in order to account for the most popular variables over the set of local splits. See Breiman (2002) and Liaw et al. (2002) for further details on this topic, and Archer and Kimes (2008) for a case of study in a real life example.

1.2.2 Optimal Classification Trees

The formulation of an Optimal Classification Tree (OCT) was firstly introduced in Bertsimas and Dunn (2017). The authors proposed two mathematical optimization problems, one of these aiming to obtain OCT with univariate splits, and another one to obtain Optimal Classification Trees with oblique splits (OCT-H). These models have proven to obtain higher accuracies than other heuristic approaches in different real-life datasets, showing that the models were able to capture more reliable data distributions.

In the rest of this section we focus on describing the OCT formulation, but before doing this, we need to fix the notation and introduce some parameters required by the method. The parameters OCT needs to be fixed before training the model are the following:

- D: Tree depth.
- N_{min}: Minimum number of observations required to be in a leaf node.
- c: Complexity cost parameter.
- ϵ: A vector formed by positive constants used to model the branches in the tree.

OCT is based on constructing a maximal tree for a given depth D, despite the fact that some of these nodes might be empty. Such a tree has $T = 2^{D+1} - 1$ nodes, which are indexed by $t = 1, \ldots, T$. The authors used the notation $p(t)$ to refer to the parent of node t, and $A(t)$ to denote the set of ancestors of t. In addition to this, they consider the left-branch ancestors, $A_L(t)$, as the set of ancestors of t whose left branch has been followed on the path from the root node to t, and analogously $A_R(t)$ for the right-branch ancestors. Moreover, as it has alraedy been pointed out, nodes are distinguished in two types:

- Branch nodes, $\tau_b = \{1, \ldots, \lfloor T/2 \rfloor\}$: nodes in which a split in the form $\omega' x < \omega_0$ is applied. Observations which satisfy this constraint follow the left branch whereas observations which do not satisfy it follow the right one.
- Leaf nodes, $\tau_l = \{\lfloor T/2 \rfloor + 1, \ldots, T\}$: nodes in which class predictions are made.

Given a training sample containing n observations with p features and a label over k possible classes, OCT is formulated according to minimize an objective function formed by two terms. The first one manages the number of misclassified observations whereas the second one regulates the tree complexity. Complexity is understood as the number of non-empty nodes. On the one hand, in order to control complexity, some binary variables are introduced for each of the branch nodes, $\delta_t \in \{0,1\},\ t \in \tau_b$. δ_t takes value one if t is a non-empty node, and zero otherwise. On the other hand, for the purpose of accounting for misclassification errors, some variables, $L_t \in \mathbb{Z}_+$, are introduced at each of the leaf nodes, $t \in \tau_l$, where L_t is equal to the number of misclassified observations in leaf t. Therefore, the authors derive the objective function of the problem as follows:

$$\frac{1}{\hat{L}} \sum_{t \in \tau_l} L_t + c \sum_{t \in \tau_b} \delta_t,$$

where c is the complexity cost parameter, and $\hat{L}$ is the baseline accuracy (the accuracy obtained predicting the most represented class for the whole dataset).

Regarding to OCT constraints, one may start by defining the univariate splits. The authors define the splitting hyperplane coefficients as binary variables, $\omega \in \{0,1\}^p$, an they impose the sum over these coefficients to be equal to one. Nevertheless, complexity plays its role in this part as well, since $\delta_t = 0$ means that no observations belong to node t, and therefore no splitting hyperplane should be built in such a node. In order to satisfy the above requirements, the following constraints are added to the problem:

$$\sum_{j=1}^{p} \omega_{tj} = \delta_t, \qquad \forall t \in \tau_b,$$

$$0 \leq \omega_{0t} \leq \delta_t, \qquad \forall t \in \tau_b.$$

These two constraints allow the model to build a univariate split in node t if and only if $\delta_t = 1$ (note that the second constraint is valid for normalized data, $x_i \in [0,1]^p$, which can be done without loss of generality).

Once splits and complexity variables are defined, the authors endow the tree with a hierarchical structure by means of the following constraints:

$$\delta_t \leq \delta_{p(t)}, \qquad \forall \in \tau_b \backslash \{1\}.$$

According to this, if a split is not applied in node t, further splits will not be allowed in its successors nodes.

Another important part of the model is tracking the observations allocation

within the tree nodes. In order to do this, a set of binary variables, $z_{it},\ i = 1,\dots,n,\ t = 1,\dots,T$, is added to the problem. These variables take value one if observation i is in node t, and zero otherwise. Moreover, to assure the proper functioning of the model, each observation must be assigned to exactly one leaf node (where classification errors are accounted for), which can be done by imposing the usual assignment constraints:

$$\sum_{t\in\tau_l} z_{it} = 1, \qquad \forall i = 1,\dots,n.$$

Furthermore, OCT fixes a minimum number of observations per leaf (N_{min}), and so as to regulate this, some binary variables, $l_t \in \{0,1\},\ l \in \tau_l$, need to be added to the problem to indicate whether leaf t contains points ($l_t = 1$), or not ($l_t = 0$). This requirement can be modeled by adding to the problem the following constraints:

$$\begin{aligned} &z_{it} \le l_t, && \forall i = 1,\dots,n,\ t \in \tau_l,\\ &\sum_{i=1}^{n} z_{it} \ge N_{min} l_t, && \forall t \in \tau_l. \end{aligned}$$

With the above constraints, leaf t is forced to contain more than N_{min} observations.

The allocation on the leaf nodes is not correctly defined yet. Observations need to satisfy the branch inequalities which lead indeed to leaf t so as to actually be located in t. For such a purpose, the authors start by pointing out the need of the following constraints:

$$\begin{aligned} &\omega_m' x_i < \omega_{0m} - M_1(1 - z_{it}), && \forall i = 1,\dots,n,\ t \in \tau_l,\ m \in A_L(t),\\ &\omega_m' x_i \ge \omega_{0m} - M_2(1 - z_{it}), && \forall i = 1,\dots,n,\ t \in \tau_l,\ m \in A_R(t), \end{aligned}$$

where M_1 and M_2 are big enough constants. Note that the first set of constraints use a strict inequality, and therefore the authors use a vector of parameters ϵ to transform these into non-strict inequalities. The way of computing this vector as well as the discussion on the values $M_1 = 1 + max(\epsilon)$, and $M_2 = 1$ is detailed in the paper. In the following we introduce the reformulation of these constraints presented by the authors:

$$\begin{aligned} &\omega_m'(x_i + \epsilon) \le \omega_{0m} - (1 + max(\epsilon))(1 - z_{it}), && \forall i = 1,\dots,n,\ t \in \tau_l,\ m \in A_L(t),\\ &\omega_m' x_i \ge \omega_{0m} - (1 - z_{it}), && \forall i = 1,\dots,n,\ t \in \tau_l,\ m \in A_R(t). \end{aligned}$$

Finally, misclassification errors must be defined. In order to do this, the authors

define first the variables $N_t \in \mathbb{Z}_+$, and $N_{st} \in \mathbb{Z}_+$, $t \in \tau_l,\ s = 1, \ldots, k$, as follows:

$$N_{st} = \frac{1}{2}\sum_{i=1}^{n}(1 + Y_{is})z_{it}, \qquad \forall s = 1, \ldots, k,\ t \in \tau_l,$$
$$N_t = \sum_{i=1}^{n} z_{it}, \qquad \forall t \in \tau_l,$$

where Y_{is}, which is derived from original data, takes value one if observation i belongs to class s, and minus one otherwise. According to this, N_t is the number of observations located in leaf t, and N_{st} is the number of observations belonging to class s inside leaf t. Hence, misclassified observations in leaf t can be calculated as the difference between N_t and the maximum value amongst the N_{st} variables. Recalling that L_t variables are being minimized at the objective function, the authors compute these differences by means of the following constraints:

$$L_t \geq N_t - N_{st} - M(1 - c_{st}), \qquad \forall s = 1, \ldots, k,\ t \in \tau_l,$$
$$L_t \leq N_t - N_{st} + Mc_{st}, \qquad \forall s = 1, \ldots, k,\ t \in \tau_l,$$

where M is a big enough constant (n is a valid value for this constant), and $c_{st} \in \{0, 1\}$ is a binary variable that takes value one if class s is predicted in leaf t, and zero otherwise. For these last variables, one more set of constraints is required, since exactly one class has to be predicted in leaf t (in case t is a non-empty leaf), which is done imposing the assingment constraints:

$$\sum_{s=1}^{k} c_{st} = l_t, \qquad \forall t \in \tau_l.$$

Gathering all the constraints together, OCT is formulated as the following Mixed Integer Problem (MIO):

$$\min \frac{1}{\hat{L}}\sum_{t\in\tau_l} L_t + c\sum_{t\in\tau_b}\delta_t \qquad \text{(OCT)}$$

$$
\begin{aligned}
\text{s.t.}\quad & \sum_{j=1}^{p} \omega_{tj} = \delta_t, && \forall t \in \tau_b,\\
& 0 \leq \omega_{0t} \leq \delta_t, && \forall t \in \tau_b,\\
& \delta_t \leq \delta_{p(t)}, && \forall \in \tau_b \backslash \{1\},\\
& \sum_{t \in \tau_l} z_{it} = 1, && \forall i = 1 \ldots, n,\\
& z_{it} \leq l_t, && \forall i = 1, \ldots, n,\ t \in \tau_l,\\
& \sum_{i=1}^{n} z_{it} \geq N_{min} l_t, && \forall t \in \tau_l,\\
& \omega'_m (x_i + \epsilon) \leq \omega_{0m} - (1 + max(\epsilon))(1 - z_{it}), && \forall i = 1, \ldots, n,\ t \in \tau_l,\ m \in A_L(t),\\
& \omega'_m x_i \geq \omega_{0m} - (1 - z_{it}), && \forall i = 1, \ldots, n,\ t \in \tau_l,\ m \in A_R(t),\\
& N_{st} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{is}) z_{it}, && \forall s = 1, \ldots, k,\ t \in \tau_l,\\
& N_t = \sum_{i=1}^{n} z_{it}, && \forall t \in \tau_l,\\
& L_t \geq N_t - N_{st} - n(1 - c_{st}), && \forall s = 1, \ldots, k,\ t \in \tau_l,\\
& L_t \leq N_t - N_{st} + n c_{st}, && \forall s = 1, \ldots, k,\ t \in \tau_l,\\
& \sum_{s=1}^{k} c_{st} = l_t, && \forall t \in \tau_l,\\
& \omega_t \in \{0,1\}^p,\ \omega_{0t} \in \mathbb{R},\ \delta_t \in \{0,1\}, && \forall t \in \tau_b,\\
& l_t \in \{0,1\},\ N_t,\ L_t \in \mathbb{R}, && \forall t \in \tau_l,\\
& N_{st} \in \mathbb{R},\ c_{st} \in \{0,1\}, && \forall s = 1, \ldots, k,\ t \in \tau_l,\\
& z_{it} \in \{0,1\}, && \forall i = 1, \ldots, n.
\end{aligned}
$$

1.3 Fitting Hyperplanes Theory

In this section we summarize some of the elements presented in Blanco et al. (2018) about fitting hyperplanes problems. The problem of locating hyperplanes with respect to a given set of point is well known in Location Theory (LT) (Schöbel (2013)). This problem is closely related to another common question in Data Analysis: to study the behavior of a given dataset with respect to a fitting body expressed with an equation of the form $f(x) = 0$, with $x = (X_1, \ldots, X_p) \in \mathbb{R}^p$. This last problem reduces to the estimation of the 'best' function f that expresses the relationship between the data or, in the jargon of LT, to the location of the surface $f(x) = 0$ that minimizes some aggregation function of the distances to these points (see Amaldi et al. (2016); Drezner et al. (2002); Dıaz-Bánez et al. (2004)). In many cases the family

of functions where f belongs to is fixed and then, the parameters defining such an *optimal* function must be determined. The family of linear functions is the most widely used. This implies that the above equation is of the form $f(x) = \omega_0 + \sum_{l=1}^{p} \omega_l X_l = 0$ for $\omega_0, \omega_1, \ldots, \omega_p \in \mathbb{R}$.

To perform such a fitting, we are given a training sample $\{x_1, \ldots, x_n\} \subset \mathbb{R}^p$, where the goal is to find $\hat{\omega}_0$ and $\hat{\omega} = (\hat{\omega}_1, \ldots, \hat{\omega}_p)$ that minimizes some measure of the deviation of the data with respect to the hyperplane they induce, $\mathcal{H} = \{z \in \mathbb{R}^p : \hat{\omega}' z + \hat{\omega}_0 = 0\}$. For a given point $x \in \mathbb{R}^p$, we define the *residual* with respect to a generic x as a mapping $\varepsilon_x : \mathbb{R}^{p+1} \to \mathbb{R}_+$, that maps any set of coefficients $(\omega_0, \omega) = (\omega_0, \omega_1, \ldots, \omega_p) \in \mathbb{R}^{p+1}$, into a measure $\varepsilon_x(\omega_0, \omega)$ that represents the deviation of the given point x from the hyperplane with those parameters. The problem of locating a hyperplane for a given set of points $\{x_1, \ldots, x_n\} \subseteq \mathbb{R}^p$ consists of finding the coefficients minimizing an aggregation function, $\Phi : \mathbb{R}^n \to \mathbb{R}$, of the residuals of all the points. Different choices for the residuals and the aggregation criteria will give, in general, different optimal values for the parameters and thus different properties for the resulting hyperplanes. This problem is not new and some of these criteria, as the minisum, minimax and some other alternatives, have been widely analyzed from a LT perspective (see Megiddo and Tamir (1983); Schöbel (1996, 1997, 1998, 2013), amongst others).

For a further analysis on the problem of locating a hyperplane to fit our training sample, minimizing different forms of measuring the residuals and their aggregation, we define the Fitting Hyperplane Problem (FHP) as the problem of finding $\hat{\omega}_0$ and $\hat{\omega}$ such that:

$$\hat{\omega}_0, \hat{\omega} \in \arg \min_{\omega_0 \in \mathbb{R},\ \omega \in \mathbb{R}^p} \Phi(\epsilon(\omega_0, \omega)), \tag{FHP}$$

where $\epsilon(\omega_0, \omega) = (\varepsilon_1(\omega_0, \omega), \ldots, \varepsilon_n(\omega_0, \omega))$ is the vector of residuals. Note that the difficulty of solving the FHP depends on both the expressions for the residuals and the aggregation criterion Φ. If Φ and ε_x are linear, the above problem becomes a linear programming problem (LP).

In the following we present a multiparametric family of functions, called *ordered median functions*, introduced in Nickel et al. (2005). In order to do this, let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ and let $\epsilon \in \mathbb{R}^n$ be the vector of residuals of all of the points in the given training sample. Thus, the *ordered median functions* are derived by means of the aggregation criteria $\Phi : \mathbb{R}^n \to \mathbb{R}_+$ defined as:

$$\Phi(\epsilon) = \sum_{i=1}^{n} \lambda_i \, \epsilon_{(i)}^{\rho}, \quad 1 \leq \rho < +\infty,$$

where $\epsilon_{(i)} \in \{\epsilon_1, \ldots, \epsilon_n\}$ is such that $\epsilon_{(1)} \leq \cdots \leq \epsilon_{(n)}$.

According to this, the choice of the λ-weights will lead to different problems, capturing many of the models proposed in the literature. Most classical models assume that the residuals are defined as the vertical distance (with respect to the last coordinate) from the points to the hyperplane:

$$\epsilon_x(\omega_0, \omega) = \left| x_p - \frac{\omega_0}{\omega_p} - \sum_{l=1}^{p-1} \frac{\omega_l}{\omega_p} x_l \right|,$$

(assuming that $\omega_p \neq 0$). Therefore, the difference between them comes from the choice of the aggregation criterion Φ. We show below how some classical methods can be accommodated to this framework.

1. The Least Sum of Squares (LSS) method, credited to Gauss (1877), is the most widely used approach to estimate the coefficients of a linear model due to its simplicity (a closed form for the optimal coefficients is obtained) and its theoretical implications for the inference over the total population. However, somehow restricting hypotheses are required in order to be applied (see, e.g., Giloni and Padberg (2002)). The LSS criterion is defined as the sum of the squares of the residuals, that is: $\Phi_{LSS}(\epsilon_1, \ldots, \epsilon_n) = \sum_{i=1}^{n} \epsilon_i^2$, where the residuals ϵ_i are given by the above expression. The reader may observe that LSS corresponds to the FHP with $\lambda = (1, \ldots, 1)$, $\rho = 2$ and ϵ the vertical distance.

2. The Least Absolute Deviation (LAD) method, introduced by Edgeworth (1887), consists of minimizing the sum of the absolute value of the vertical residuals. Therefore, $\Phi_{LAD}(\epsilon_1, \ldots, \epsilon_n) = \sum_{i=1}^{n} |\epsilon_i|$. Note that LAD corresponds to the FHP for $\lambda^t = (1, \ldots, 1)$ and $\rho = 1$.

3. The Least Quantile of Squares (LQS), introduced by Bertsimas and Mazumder (2014), is a generalization of the Least Median of Squares (LMS) introduced by Hampel (1975). It also considers vertical distances as residuals, but they are aggregated to minimize the r-quantile of its distribution (r ranges in $\{1, \ldots, n\}$). Hence, $\Phi_{LQS}(\epsilon_1, \ldots, \epsilon_n) = r - \text{quantile}(\epsilon_1^2, \ldots, \epsilon_n^2) := \epsilon_{(r)}^2$.

This method also fits to the general form of the aggregation criteria considered in this paper. In this case, the LQS hyperplane can be obtained for $\rho = 2$ and $\lambda = (\overbrace{0, \ldots, 0}^{(r-1)}, 1, \overbrace{0, \ldots, 0}^{(n-r)})$. (Observe that LMS hyperplane is also obtained within the same scheme when $p = 2$ and $\lambda = (\overbrace{0, \ldots, 0}^{\lfloor \frac{n}{2} \rfloor}, 1, \overbrace{0, \ldots, 0}^{\lfloor \frac{n}{2} \rfloor})$).

4. The Least Trimmed Sum of Squares (LTS) method was introduced by Rousseeuw Rousseeuw (1984) as a robust alternative to the LSS method, in that it has a high breakdown point. Recall that, intuitively, the *breakdown point* of an estimator is the proportion of incorrect observations (e.g., arbitrarily large observations) an estimator can handle before giving an incorrect (e.g., arbitrarily large) result. With our notation, it corresponds to choose again as residuals the vertical distance, $\rho = 2$, and the aggregation criterion $\Phi_{LTS}(\epsilon_1, \ldots, \epsilon_n) = \sum_{i=1}^{h} \epsilon_{(i)}^2$ where $\epsilon_{(i)} \in \{\epsilon_1, \ldots, \epsilon_n\}$ with $\epsilon_{(i)} \leq \epsilon_{(i+1)}$ for $i = 1, \ldots, n-1$, and $h \in \{1, \ldots, n\}$. The most common choice for h is $\lfloor \frac{n}{2} \rfloor$, considering the best 50% square residuals.

Going back to the general agregation and residual function definitions, on the one hand we have that the function Φ is invariant against permutations of its components and, for non negative lambda weights, a monotone function, ensuring that the ordering of the individual residuals do not affect the overall goodness of the fitting. Moreover, it also implies that a componentwise smaller vector of residuals gives rise to a more accurate fitting. On the other hand, the natural implication of the assumption made about the definition of residuals is that, as expected, the response (projection) of a point on a given hyperplane differs from the classical evaluation. In this setting the response is the closest point, with respect to the distance D, to the hyperplane $\mathcal{H}$, and for a given point $x \in \mathbb{R}^p$ this is, recalling from Mangasarian (1999), $\epsilon_x = \dfrac{|\omega' x + \omega_0|}{\|\omega\|_*}$, where $\|\cdot\|_*$ is the dual norm of the one inducing distance D. We should remark at this point that the standard residual (vertical distance) is a distance measure that is not induced by a norm, however its expression can be written in a analogous form and so it fits to the shape of the distances that are considered in this FHP.

Gathering all these elements, and considering $\lambda \in \mathbb{R}^n$, such that $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$, the FHP is equivalent to the following Mathematical Programming problem (Blanco et al. (2018)):

$$
\begin{aligned}
\min \ & \sum_{j=1}^{n} u_k + \sum_{i=1}^{n} v_i & & \\
\text{s.t. } & u_k + v_i \geq \lambda_k e_i, & & \forall i, k = 1, \ldots, n, \\
& e_i \geq \epsilon_{x_i}(\omega_0, \omega), & & \forall i = 1, \ldots, n, \\
& e_i, u_i, v_i \in \mathbb{R}_+, & & \forall i = 1, \ldots, n, \\
& \omega_0 \in \mathbb{R}, \omega \in \mathbb{R}^p. & &
\end{aligned}
$$

According to this, the first set of constraints control the order over the residuals

meanwhile the second one ensures proper residual definition. Hence, the distance involved in such a problem will determine whether we obtain a LP (as for instance when using vertical distances, ℓ_1 or ℓ_∞ norms) or a NLP (as for instance when using the Euclidean norm). To conlcude this section, observe that when the points in the dataset lie exactly on a hyperplane, $\mathcal{H}$, this hyperplane is always optimal for all versions of the FHP, although for some specific choices of λ the solution may not be unique and different hyperplanes may be alternative optimal.

1.4 Contributions of this thesis

In this thesis we analyze different classical Data Science problems through a Location Theory perspective. In particular, we focus on locating hyperplanes to address the problems.

Chapter 2 is based on the paper Blanco et al. (2020b). In this chapter we present a multiclass extension of the SVM methodology. In contrast to other SVM multiclass approaches, we propose some MIP and MINLP formulations which maximize the minimum of the margins amongst a set of hyperplanes. Furthermore, our methodology is based on a novel idea consisting on dividing the p feature space, by means of locating j hyperplanes, into a set of cells that are unequivocally identified with a pattern of signs over a j-tuple. Once the hyperplanes are located, each of the cells is assigned to the most represented class amongst its obervations. Therefore, the decision rule for out-of-sample observations is nothing but predicting the class of the cell in which observations lie in, or the class of the closest non-empty cell. We propose four different models depending on the norm used to measure the margins, ℓ_1 or ℓ_2, and depending on whether using a bounded or an unbounded error meassure. Moreover, we prove that the kernel trick can be used in all the models. Finally, we report some computational experiment results in which we show the effectiveness of our approaches.

Afterwards studying a multiclass classification problem by means of locating m hyperplanes, we study in Chapter 3 the problem of locating m hyperplanes minimizing a distance based error function from observations to the hyperplanes. Although we study a very general problem, allowing one to use many different distances measures and error functions, particular cases result into usual Data Science problems. For instance, when fixing $m = 1$ and considering the vertical distance, our problem turns into classical linear regression. Therefore, locating m hyperplanes when using the vertical distance can be seen as a m-extension to the classical linear regression problem.

Chapter 3 is based on the paper Blanco et al. (2021a). In this work, two approaches are presented to solve the above mentioned problems. On the one hand, exact prob-

lem formulations are presented by means of some MIP and MINLP. On the other hand, some set partitioning formulations are presented as well as a column generation procedure to solve them. Moreover, convergence results are proven under certain conditions, and some heuristics are also developed so as to speed up the solution processes. Lastly, a discussion of the presented methods performance over an extensive battery of computational experiments is reported.

Whereas Chapters 2 and 3 consider a set of m hyperplanes, Chapter 4, which is based on the work Blanco et al. (2020a), focuses on locating a single hyperplane in a binary classification context. This hyperplane is located following a SVM classification principle where some noise might exist within the observation labels set. The motivation of this work comes from different real life situations in which the reliability of the label set in the training sample is not completely truthful. In order to approach this problem, we present three models which combine SVM and cluster ideas. The goal of these models is to create two clusters, as homogeneous as possible, which are separated by a hyperplane, and at the same time the margin betweem them is maximized. A SVM hyperplane separates observations with respect to their classes, nevertheless, we are able to separate observations with respect to the cluters since our models permit observations to be relabeled, i.e., to be treated as if they belong to their opposite class. These methods obtain much higher accuracies than standard SVM when some label noise is added to the training sample, as can be seen in the computational experiments of the work.

Optimal Classification Trees have been studied when using splits that do not maximize the separating hyperplanes margin and do not take into account distance based errors. In Chapter 5, which is based on the paper Blanco et al. (2021c), we propose a formulation of an OCT with SVM-based splits, that we call OCTSVM, for binary classification problems. Moreover, in our solution we take advantage of the relabel formulation presented in Chapter 4, and hence our model is also robust against label noise on the training sample. The computational results presented in this chapter show how OCTSVM outperforms OCT, OCT-H and CART methodologies in both noisy label training samples and normal training samples.

The multiclass extension of the OCTSVM, that we call MOCTSVM, is later on introduced in Chapter 6. This chapter is based on the work by Blanco et al. (2021b). The tools developed in Chapter 4 are again essential to allow us to formulate the MOCTSVM. This is due to the fact that SVM splits are designed to separate two classes, and therefore we permit the model to create SVM-splits by relabeling observations over two fictitious classes that are separated in the branch nodes, until observations reach the leaf nodes were misclassification errors amongst the original labels are measured. This method is rather similar to the multiclass SVM approach presented in Chapter 2, nevertheless, MOCTSVM follows a hierarchical structure

when locating the m hyperplanes. According to this, when looking at the global configuration of the hyperplanes in the p-features space, some parts of themselves do not play any action in the decision rule, what can lead to a lower number of cells in a classification problem. This does not necessarily have an impact on the predictive power of the model, but the interpretability of the solution is often easier when the number of cells is low. Finally, we conclude this chapter with some computational experiment results which show that MOCTSVM reports more accurate results than CART, OCT and OCT-H in a battery of real life datasets.

Chapter 2

Multiclass Support Vector Machines

In this chapter we present a novel SVM-based approach to construct multiclass classifiers by means of arrangements of hyperplanes. We propose different mixed integer (linear and non linear) programming formulations for the problem using extensions of widely used measures for misclassifying observations where the *kernel trick* can be adapted to be applicable. Some dimensionality reductions and variable fixing strategies are also developed for these models. An extensive battery of experiments has been run which reveal the powerfulness of our proposal as compared with other previously proposed methodologies.

2.1 Introduction

SVM has proven to be a very useful tool in a wide range of real world applications (see e.g. Bahlmann et al. (2002); Harris (2013); Majid et al. (2014); Radhimeenakshi (2016)). Most of the SVM literature concentrates on binary classification where several extensions are available. One can use different measures for the separation between classes (see e.g., Blanco et al. (2020d); Ikeda and Murata (2005)), select important features (Labbé et al. (2019)), apply regularization strategies (López et al. (2018); Marín et al. (2021)), etc. However, the analysis of SVM-based methods for datasets with more than two classes has been, from our point of view, only partially investigated. Given a training sample $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times \{1, \ldots, k\}$, the multiclass ($k > 2$) SVM consists of constructing a decision rule able to accurately classify out-of-sample observations.

Different multiclass classification techniques that take advantage of the SVM methods for binary classification can be found in the literature. The most popular multiclass SVM-based approaches are OVA and OVO. The former, OVA, computes, for each class $r \in \{1, \ldots, k\}$, a binary SVM classifier labeling the observations as 1, if the observation is in the class r and -1 otherwise. The process is repeated for all classes (k times), and then each observation is classified into the class whose constructed hyperplane is the furthest from it in the positive halfspace. In the OVO approach, classes are separated with $\binom{k}{2}$ hyperplanes using one hyperplane for each pair of classes, where the decision rule comes from a voting strategy in which the most represented class amongst votes becomes the class predicted. OVA and OVO inherit most of the good properties of binary SVM. In spite of that, they are not able to correctly classify datasets where separated clouds of observations may belong to the same class (and thus are given the same label) when a linear kernel is used. Another popular method is the directed acyclic graph SVM, DAGSVM (Platt et al. (1999)). In this technique, although the decision rule involves the same hyperplanes built with the OVO approach, it is not given by a unique voting strategy but for a sequential number of votings in which the most unlikely class is removed until

only one class remains. In addition, apart from OVA and OVO, there are some other methods based on decomposing the original multiclass problem into several binary classification ones. In particular, in (Allwein et al. (2000)) and (Dietterich and Bakiri (1994)), this decomposition is based on the construction of a coding matrix that determines the pairs of classes that will be used to build the separating hyperplanes. Alternatively, other methods such as CS (Crammer and Singer (2001)), WW (Weston and Watkins (1998)) or LLW (Lee et al. (2004)), do not address the classification problem sequentially but as a whole considering all the classes within the same optimization model. Obviously, this seems to be the correct approach. In particular, in WW, k hyperplanes are used to separate the k classes, each hyperplane separating one class from the others, using $k-1$ misclassification errors for each observation. The same separating idea, is applied in CS but reducing the number of misclassification errors for each observation to a unique value. In LLW, a different error measure is proposed to cast the Bayes classification rule into the SVM problem implying theoretical statistical properties in the obtained classifier. These properties cannot be ensured in WW or CS.

We can also find a quadratic extension based on LLW proposed by Guermeur and Monfrini (2011). Finally, van den Burg and Groenen (2016) propose a multiclass SVM-based approach, GenSVM, in which the classification boundaries for a problem with k classes are obtained in a $(k-1)$-dimensional space using a simplex encoding. Some of these methods have become popular and are implemented in most software packages in Machine Learning as `e1071` (Meyer et al. (2019)), `scikit-learn` (Abraham et al. (2014)) or Lauer and Guermeur (2011). Nevertheless, as far as we are concerned, none of the existing multiclass SVM methods keeps the essence of binary SVM which stems from finding a globally optimal partition of the feature space.

In this chapter we propose a novel approach to handle multiclass classification extending the paradigm of binary SVM classifiers. In particular, our method finds a polyhedral partition of the feature space and an assignment of classes to the *cells* of the partition, by maximizing the separation between classes and minimizing two intuitive misclassification errors. Obviously, as in standard SVM, we can also account in different ways the misclassification errors (hinge or ramp-based losses). For bi-class instances, and using a single separating hyperplane, our method coincides with the standard SVM. Nevertheless, even for 2-classes datasets, new alternatives appear if more than one hyperplane is permited to separate the data. In particular, our approach allows one to generalize the polyhedral conic classifiers presented in (Bagirov et al. (2013)).

Apart from justifying the rationale of our method, we also propose different Mathematical Programming formulations in order to solve the resulting optimization problems. These formulations belong to the family of Mixed Integer (Linear and Non

Linear) Programming (MILP and MINLP) problems, in which the nonlinearities come from the representation of the Euclidean distance margin between classes, that can be modeled as a set of second order cone constraints (see Blanco et al. (2014)). This type of constraints can be handled nowadays by any of the most popular off-the-shelf optimization solvers (CPLEX, Gurobi, XPress, SCIP, ...).

These models also have a combinatorial nature induced by the correct allocation of labels to cells. Therefore, they require to use some binary variables. This approach is not new and recently, a few attempts have been proposed for different classification problems using discrete optimization tools. For instance, Üney and Türkay (2006) construct classification hyperboxes for multiclass classification, Bennett et al. (1999) provide formulations for SVM with unlabelled data (semi-supervised SVM), and Ghaddar and Naoum-Sawaya (2018), López et al. (2018) and Labbé et al. (2019), mixed integer linear programming tools for feature selection in SVM. Handling a large number of binary variables in the models may become an inconvenient when trying to compute classifiers for medium to large size instances. This inconvenience is alleviated with some preprocessing and dimensionality reduction techniques that are also introduced.

In case the data are, by nature, nonlinearly separable, in classical SVM one can apply the so-called kernel trick to project the data out onto a higher dimensional space where the linear separation has a better performance. The key point is that one does not need to know neither the dimension of the final space nor the specific transformation that is applied to the data: the resulting Mathematical Programming problem is in the same space as the original one. Here, we show that the kernel trick can be extended to our framework and therefore, it also allows us to find nonlinear classifiers with this methodology.

To assess the validity of our method we have performed a battery of computational tests on two different families of data. We have tested our method against some well known multiclass SVM classifiers (OVO, CS, WW and LLW) on 7 real databases. Moreover, we also report results on synthetic datasets specially tailored to capture the difficulty of multiclass supervised classification. In all cases, our methods give results similar or superior to those provided for the other methods. In particular, for the synthetic data instances the improvement in accuracy on the test samples are remarkable (see Table 2.3).

The rest of the chapter is organized as follows. In sections 2.2 and 2.3 we describe and set up the elements of the problem to be considered. Afterwards, we introduce a MINLP formulation for our model. Alternatively, we also present a linear version, which is obtained whenever the margins are measured with the ℓ_1-norm. A discussion on the extension, with very few modifications, of the previous models to the Ramp Loss versions is included as well. In Subsection 2.3.2 we discuss

how an analogous to the kernel trick can be extended to be applied in this model. Section 2.4 describes some heuristic strategies, preprocessing and dimensionality reductions to obtain good quality initial solutions of the MINLP. Finally, in section 2.5 we report our computational results on different real and synthetic datasets, and compare our method with the standard ones for multiclass SVM.

2.2 Preliminaries

In this section we introduce the problem under study. Given a training sample $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times \{1, \ldots, k\}$ the goal of supervised classification is to find a decision rule to assign labels to data, in order to be applied to out-of-sample data. We assume that a given number, m, of hyperplanes in $\mathbb{R}^p$ have to be built to obtain a subdivision of this space into full dimension polyhedral regions that we shall denote as *cells*. (Here, we would like to mention that the term *cell* stands for a nonempty intersection of the semispaces induced by the hyperplanes in the considered family). Let us denote by $\mathcal{H}_1, \ldots, \mathcal{H}_m$ the hyperplanes to be found, which are in the form $\mathcal{H}_r = \{z \in \mathbb{R}^p : \omega_r' z + \omega_{r0} = 0\}$ for some $\omega_r \in \mathbb{R}^p, \omega_{r0} \in \mathbb{R}$, for $r = 1, \ldots, m$. Each cell induced with such an arrangement of hyperplanes will be then assigned to a label in $\{1, \ldots, k\}$. In Figure 2.1 we illustrate a subdivision of $\mathbb{R}^2$ induced by 2 hyperplanes and the labels assigned to each cell. In the left figure, we represent the observations, highlighting the classes with different symbols (stars, circles and squares). In the right figure, two hyperplanes which induce 4 cells are constructed to separate the three classes. Each cell is assigned to a class (north $\rightarrow$ circles, south $\rightarrow$ stars, east $\rightarrow$ stars and west $\rightarrow$ squares). In this example the subdivision in cells and the assignment of labels reaches a perfect classification on the given observations.

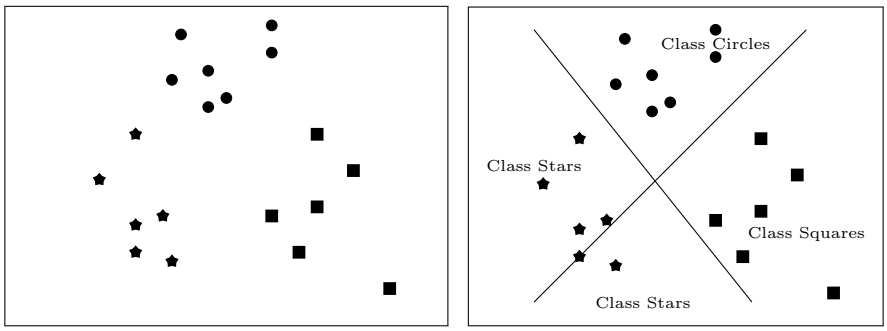


Figure 2.1: Illustration of a subdivision induced by 2 hyperplanes in $\mathbb{R}^2$.

From the above, we would like to construct an arrangement of m hyperplanes, $\mathbb{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_m\}$, determined by $\omega_1, \ldots, \omega_m \in \mathbb{R}^p$ (coefficients) and $\omega_{10}, \ldots, \omega_{m0} \in$

$\mathbb{R}$ (intercepts) and a decision rule that assigns a single label to each one of the cells in the subdivision of the space induced by such an arrangement. We would like to point out that each cell in the subdivision can be univocally identified with a $\{-1,+1\}$-vector in $\mathbb{R}^m$: the ℓ-component of that vector represents the side (positive or negative) with respect to the hyperplane $\mathcal{H}_\ell$ where that cell lies in.

Definition 2.1 (Suitable Assignment)**.** *Given a subdivision $\mathcal{C}$ of $\mathbb{R}^p$ into cells induced by the arrangement of hyperplanes $\mathbb{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_m\}$ in $\mathbb{R}^p$, a function $g : \{-1,1\}^m \to \{1,\ldots,k\}$ is said a* suitable assignment*, if g univocally maps cells (equivalently, sign-patterns) to labels in $\{1,\ldots,k\}$.*

Observe that a suitable assignment, g, allows us to classify any observation $x \in \mathbb{R}^p$ within the set of classes $\{1,\ldots,k\}$, as follows:

1. Identify x with a sign-pattern: $\mathrm{s}(x) = (\mathrm{s}_1(x), \ldots, \mathrm{s}_m(x)) \in \{-1,+1\}^m$, where $\mathrm{s}_r(x) = \mathrm{sign}(\omega_r' x + \omega_{r0})$ for $r = 1,\ldots,m$.

2. Apply the function g to the sign-patterns: $\widehat{y}(x) = g(\mathrm{s}(x)) \in \{1,\ldots,k\}$, is the *predicted* label of x.

The quality of the decision rule is based, on comparing predictions and actual labels on a training sample, but also on maximally separating the classes in order to find good predictions and avoid undesired overfitting.

SVM is a particular case of our approach for classifying two-class datasets if $m = 1$, i.e., a single hyperplane to subdivide the feature space is used. In such a case, signs are in $\{-1,1\}$ and classes in $\{1,2\}$, so whenever there are observations in both classes, the assignment is one-to-one. However, even for biclass instances, if more than one hyperplane is used, one may find better classifiers (we illustrate this behavior with the dataset `2C4N` of our computational experiments in Table 2.3). In Figure 2.2, left-and-right, we draw the same dataset of labeled (red and blue) observations and the result of applying a standard SVM (left) and our method with 2 hyperplanes. In that picture one may see that not only the misclassification errors are smaller with two hyperplanes, as expected, but also the separation between classes is larger, improving the predictive power of the classifier.

The rationale of our approach is particularly adequate for datasets in which there are several separated "clouds" of observations that belong to the same class. In Figure 2.3, we show two different instances in which, again, the colors indicate the class of the observations. The classes in both instances cannot be appropriately separated using any of the available linear SVM-based methods in the literature since they are based on subdividing the space on class-connected regions. However, we are able to perfectly separate the classes using 5 hyperplanes.

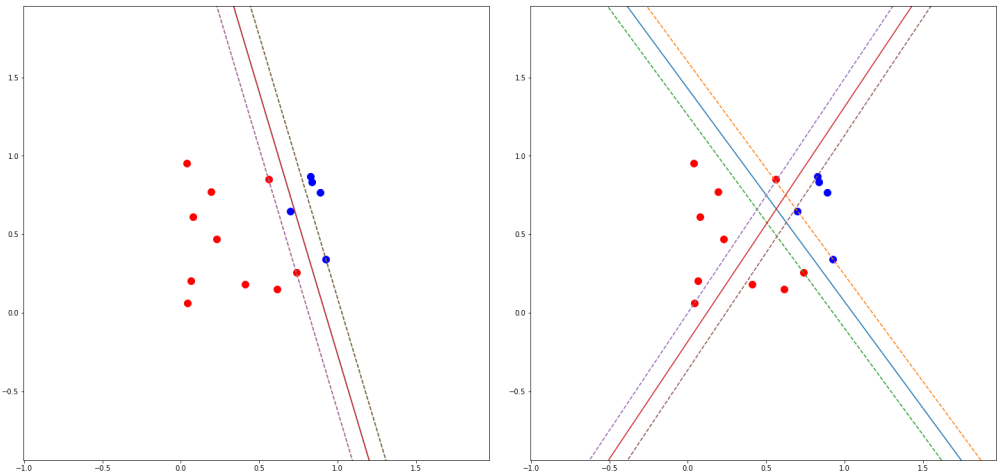


Figure 2.2: Standard SVM (left) and our approach with 2 hyperplanes (right).

In Figure 2.4 we compare our approach and the OVO approach in an instance with 24 observations. In the left figure we show the result of separating the classes with four hyperplanes, reaching a perfect classification on the training sample. In the right figure we show the best linear OVO classifier, in which only 66% of the data were correctly classified. We would also like to highlight that, although nonlinear SVM-approaches may separate the data more conveniently, our approach may help to avoid using kernels and ease the interpretation of the results.

Different alternatives could be admissible to justify the rationale of the multiclass classifiers in our framework. To simplify the presentation, we will concentrate on two different models which share the same paradigm but differ in the way they account for misclassification errors. Recall that in SVM-based methods, two criteria are *simultaneously* optimized when constructing a classifier. On the one hand, a measure of the quality of the decision rule on out-of-sample observations, based on finding a maximum separation between classes; and on the other hand a measure of the misclassification errors for the training set of observations. Both criteria are adequately weighted in order to find a good compromise between the two goals.

In what follows we describe how similar measures can be defined in our multiclass classification framework and the way we account them for.

2.2.1 Separation between classes

Separation between classes will be measured as it is usual in SVM-based methods. Let $\omega_1, \ldots, \omega_m \in \mathbb{R}^p$ and $\omega_{10}, \ldots, \omega_{m0} \in \mathbb{R}$ be the coefficients and intercepts of a set of hyperplanes. The distance induced by a norm $\|\cdot\|$ between the shifted hyperplanes $\mathcal{H}_r^+ = \{z \in \mathbb{R}^p : \omega_r' z + \omega_{r0} = 1\}$ and $\mathcal{H}_r^- = \{z \in \mathbb{R}^p : \omega_r' z + \omega_{r0} = -1\}$ is given by

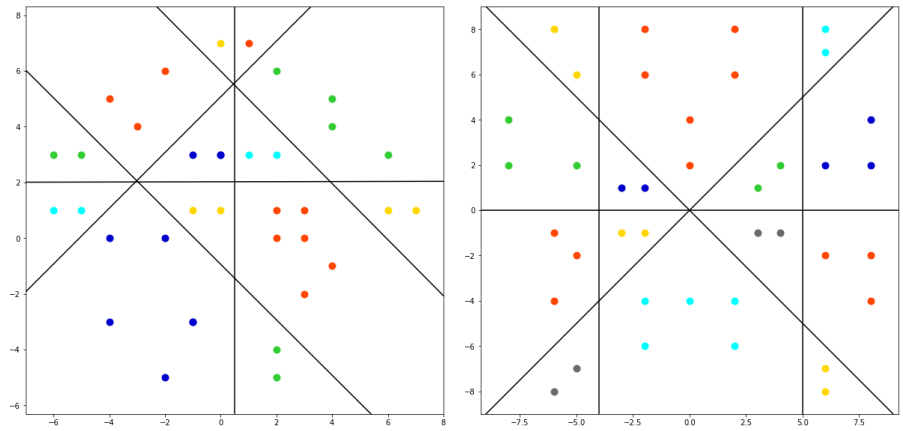


Figure 2.3: A 5-classes instance classified with our approach using 5 hyperplanes (left) and a 6-classes instance classified with our approach using 5 hyperplanes (right).

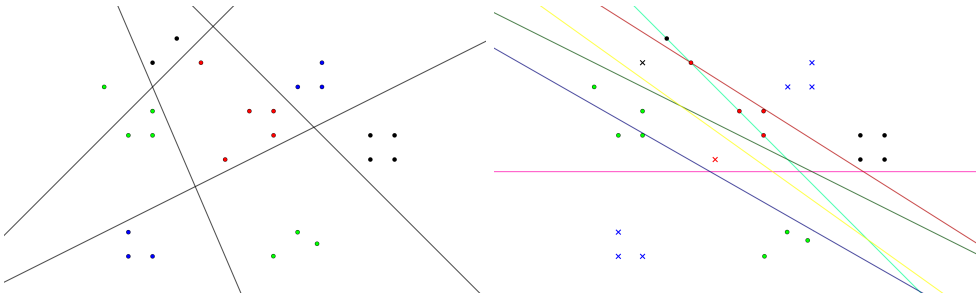


Figure 2.4: A 4-classes instance classified with our approach using 4 hypeplanes (left) and the same intance classified using the OVO SVM approach (right).

$\frac{2}{\|\omega_r\|_*}$, where $\|\cdot\|$ is a given norm in $\mathbb{R}^p$ and $\|\cdot\|_*$ is its dual norm (see Mangasarian (1999)). Unless explicitly mentioned, we will consider that $\|\cdot\|$ is the Euclidean norm which dual is also the Euclidean norm.

Hence, in order to find globally optimal hyperplanes with maximum separation, we maximize the minimum separation between classes, that is $\min\left\{\frac{2}{\|\omega_1\|}, \ldots, \frac{2}{\|\omega_m\|}\right\}$. This measure will conveniently keep the minimum separation between classes as large as possible. Observe that finding the maximum min-separation is equivalent to minimize $\max\{\frac{1}{2}\|\omega_1\|^2, \ldots, \frac{1}{2}\|\omega_m\|^2\}$. For a given arrangement of hyperplanes, $\mathbb{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_m\}$, we will denote by $h_H(\mathcal{H}_1, \ldots, \mathcal{H}_m) = \max\{\frac{1}{2}\|\omega_1\|^2, \ldots, \frac{1}{2}\|\omega_m\|^2\}$.

We note in passing that different criteria could have been used to model the separation between classes. For instance, one may consider to maximize the summation of all separations namely $\sum_{r=1}^{m} \frac{2}{\|\omega_r\|^2}$ or the inverse of that summation, namely

$\frac{2}{\sum_{r=1}^{m}\|\omega_r\|^2}$. However, although mathematically possible, these approaches do not capture the original concept in classical SVM and we have left them to be developed by the interested reader.

2.2.2 Misclassification errors

The performance of a classifier on the training set is usually measured with some function of the misclassification errors. Classical SVM with hinge-loss errors use, for non well-classified observations, a penalty proportional to the distance to the side in which they would have been well-classified. Then the overall sum of these errors is minimized. We extend the notion of hinge-loss errors to the multiclass setting as follows.

Let $\mathbb{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_m\}$ be an arrangement of hyperplanes and (x, y) a pair observation (x), label (y), with $\mathrm{s}(x) = (\mathrm{s}_1(x), \ldots, \mathrm{s}_m(x))$ being the sign-pattern of x with respect to the hyperplanes in $\mathbb{H}$. Let $g : \{-1, 1\}^m \to \{1, \ldots, k\}$ be a suitable assignment. We denote by $\mathrm{t}(x) = (\mathrm{t}_1(x), \ldots, \mathrm{t}_m(x))$ the signs of the closest cell to x whose class by g is y. We will say that (x, y) is *wrong-classified* with respect to $\mathcal{H}_r$ if $\mathrm{s}_r(x) \neq \mathrm{t}_r(x)$, otherwise it is said that (x, y) is *well-classified*.

In what follows we describe the different error measures (misclassification errors due to different causes) that will be considered for x in order to construct an optimal decision rule.

Definition 2.2 (Multiclass In-Margin Hinge-Loss)**.** *The multiclass in-margin hinge-loss for (x, y) with respect to the hyperplane $\mathcal{H}_r$ is given as:*

$$h_I\Big(x, y, \mathcal{H}_r\Big) = \begin{cases} \max\{0, 1 - \mathrm{s}_r(x) \cdot (\omega_r' x + \omega_{r0})\} & \text{if } x \text{ is well classified through } \mathcal{H}_r, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that h_I models the error due to observations that although adequately classified with respect to $\mathcal{H}_r$, belong to the margin between the shifted hyperplanes $\mathcal{H}_r^+$ and $\mathcal{H}_r^-$. These errors will be zero if the observation is wrong-classified, or if it is well-classified and does not belong to the margin induced by the r-th hyperplane.

Definition 2.3 (Multiclass Out-Margin Hinge-Loss)**.** *The multiclass out-margin hinge-loss for (x, y) with respect to the hyperplane $\mathcal{H}_r$ is given as:*

$$h_O\Big(x, y, \mathcal{H}_r\Big) = \begin{cases} 1 - \mathrm{t}_r(x) \cdot (\omega_r' x + \omega_{r0}) & \text{if } x \text{ is not well classified through } \mathcal{H}_r, \\ 0 & \text{otherwise.} \end{cases}$$

h_O measures, for wrong-classified observations, how far they are from being well-classified. This error is zero whenever an observation is well-classified. Note that if an observation, besides being wrong-classified, belongs to the margin between $\mathcal{H}_r^+$

and $\mathcal{H}_r^-$, then only h_O should be accounted for. In Figure 2.5 we illustrate the differences between the two types of losses.

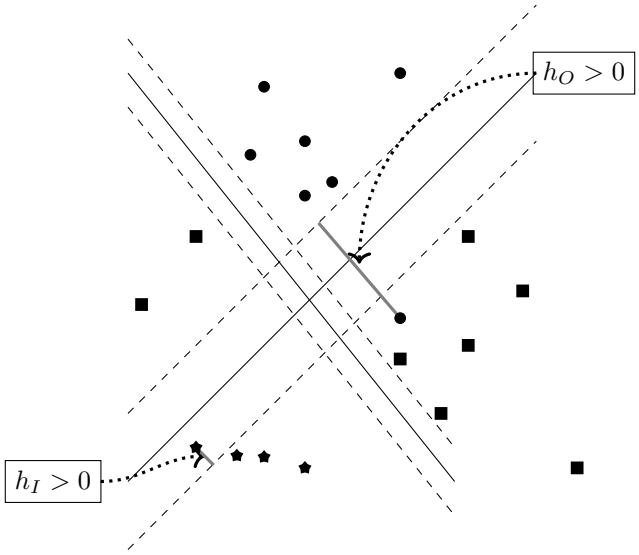


Figure 2.5: Illustration of the error measures considered in our approach.

2.3 Mathematical Programming formulations

In this section we describe the mathematical optimization models that we propose for the multiclass classification problem. Using the notation introduced in previous sections, the problem can be mathematically stated as follows:

$$\min h_H(\mathcal{H}_1, \ldots, \mathcal{H}_m) + c_1 \sum_{i=1}^{n} \sum_{r=1}^{m} h_I\Big(x_i, y_i, \mathcal{H}_r\Big) + c_2 \sum_{i=1}^{n} \sum_{r=1}^{m} h_O\Big(x_i, y_i, \mathcal{H}_r\Big) \quad (2.1)$$

$$\text{s.t. } \mathcal{H}_r \text{ is a hyperplane in } \mathbb{R}^p, \text{ for } r = 1, \ldots, m,$$

where c_1 and c_2 are parameters which model the *cost* of misclassified and strip-related errors. Usually these constants will be considered equal, nevertheless, in practice analyzing different values for them might lead to better results on predictions. A case of interest results considering $c_2 = mc_1$, i.e., the unitary cost of misclassification errors caused by out-margin observations is m times the unitary cost caused by in-margin observations. This method gives a larger penalty to wrongly classified observations, avoiding the calibration of a larger number of parameters.

Observe that the problem above consists of finding the arrangement of hyperplanes minimizing a combination of the three quality measures described in the previous section: 1) the maximum margin between classes, 2) the overall sums of the in-margin errors and 3) the out-margin misclassification errors. In what follows,

we describe how the above problem can be rewritten as a mixed integer non linear programming problem by means of adequate decision variables and constraints. Furthermore, the proposed model will consist of a set of continuous and binary variables, a linear objective function, and a set of linear and second order cone constraints. This form allows us to push the model to a commercial solver.

First, we describe the variables and constraints needed to model the first term in the objective function. We consider the continuous variables $\omega_r \in \mathbb{R}^p$ and $\omega_{r0} \in \mathbb{R}$ to represent the coefficients and intercept of the hyperplane $\mathcal{H}_r$, for $r = 1, \ldots, m$. Since there is no distinction between hyperplanes, we can assume, without loss of generality that they are non-decreasingly sorted with respect to the norms of their coefficients, i.e., $\|\omega_1\| \geq \|\omega_2\| \geq \cdots \geq \|\omega_m\|$. Then, it is straightforward to see that the term $h_H(\mathcal{H}_1, \ldots, \mathcal{H}_m)$ can be replaced in the objective function by $\frac{1}{2}\|\omega_1\|^2$, once the following set of constraints is included in the model:

$$\frac{1}{2}\|\omega_{r-1}\|^2 \geq \frac{1}{2}\|\omega_r\|^2, \qquad \forall r = 2, \ldots, m. \tag{2.2}$$

For the second term, the in-margin misclassification error, $h_I\Big(x_i, y_i, \mathcal{H}_r\Big)$, corresponding to the observation (x_i, y_i) will be identified with the continuous variable $e_{ir} \geq 0$, for $i = 1, \ldots, n$, $r = 1, \ldots, m$. Observe that to properly determine these errors, one has to determine whether the observation x_i is well-classified or not with respect to the rth hyperplane. In order to do that we need to introduce some binary variables. First, we consider the two following sets of binary variables:

$$t_{ir} = \begin{cases} 1 & \text{if } \omega_r' x_i + \omega_{r0} \geq 0, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{and} \qquad z_{is} = \begin{cases} 1 & \text{if } i \text{ is assigned to class } s, \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, n$, $r = 1, \ldots, m$, $s = 1, \ldots, k$. The t-variables model the sign-pattern of the observations, while the z-variables give the allocation profile of observations to classes. As mentioned above, the classification rule is based on assigning sign-patterns to classes.

The adequate definition of the t-variables is assured with the following constraints:

$$\omega_r' x_i + \omega_{r_0} \geq -T(1 - t_{ir}), \qquad \forall i = 1, \ldots, n,\ r = 1, \ldots, m, \tag{2.3}$$

$$\omega_r' x_i + \omega_{r_0} \leq T t_{ir}, \qquad \forall i = 1, \ldots, n,\ r = 1, \ldots, m, \tag{2.4}$$

where T is a big enough constant. Observe that T can be accurately estimated based on the data set under consideration.

Furthermore, the following constraints assure the adequate relationships between

the variables:

$$\sum_{s=1}^{k} z_{is} = 1, \qquad \forall i = 1, \ldots, n, \tag{2.5}$$

$$\|z_i - z_j\|_1 \leq 2\|t_i - t_j\|_1, \qquad \forall i = 1, \ldots, n, \jmath = 1, \ldots, n. \tag{2.6}$$

Observe that (2.5) enforces that a single class is assigned to each observation while (2.6) assures that the assignments of two observations must coincide if their sign-patterns are the same. Additionally, the set of z-variables determines whether an observation is well-classified. Indeed, let $\delta_i \in \{0,1\}^k$ be defined as $\delta_{is} = 1$ if $y_i = s$ and 0 otherwise. (Observe that δ_i is the binary encoding of the class of the ith observation.) Then, $\xi_i = \frac{1}{2}\|z_i - \delta_i\|_1 \in \{0,1\}$ assumes the value zero if and only if the observation i is well-classified, i.e.,

$$\xi_i = \begin{cases} 0 & \text{if } i \text{ is well-classified,} \\ 1 & \text{otherwise.} \end{cases}$$

Now, we will model whether the ith observation is well-classified or not, with respect to the rth hyperplane. Observe that the measure of how far is a wrong-classified observation from being well-classified, needs a further analysis. One may have a wrong-classified observation and several training observations in its same class. We assume that the error for this observation is the misclassification error with respect to the closest cell for which there are well-classified observations in its class. Thus, we need to model the decision on the well-classified *representative* observation for a wrong-classified observation. In Figure 2.6, we illustrate this type of misclassification errors. The observation x_i is wrong-classified but the misclassification error of x_i, in case x_j is chosen as its representative (well-classified) observation, is 0 with respect to hyperplane $\mathcal{H}_1$ (note that both x_i and x_j are in the same side of $\mathcal{H}_1$), whereas the misclassification error with respect to $\mathcal{H}_2$ is h. Observe that h is the distance between x_i and the shifted hyperplane defining the halfspace where x_j lies in. We consider the following set of binary variables:

$$h_{ij} = \begin{cases} 1 & \text{if } x_j, \text{ which is well classified and verifies } y_j = y_i, \text{ is the representative of } x_i \\ & \text{in its closest cell through hyperplanes,} \\ 0 & \text{otherwise.} \end{cases}$$

These variables require to impose the following constraints:

$$\sum_{\substack{j=1:\\ y_i=y_j}}^{n} h_{ij} = 1, \qquad \forall i = 1, \ldots, n, \tag{2.7}$$

$$\xi_j + h_{ij} \leq 1, \qquad \forall i, j = 1, \ldots, n, \ (y_i = y_j), \tag{2.8}$$

$$h_{ii} = 1 - \xi_i, \qquad \forall i = 1, \ldots, n. \tag{2.9}$$

The first set of constraints, (2.7), imposes a single assignment between observations belonging to the same class. Constraints (2.8) avoid choosing wrong-classified representative observations. The set of constraints (2.9) enforces well-classified observations to be represented by themselves.

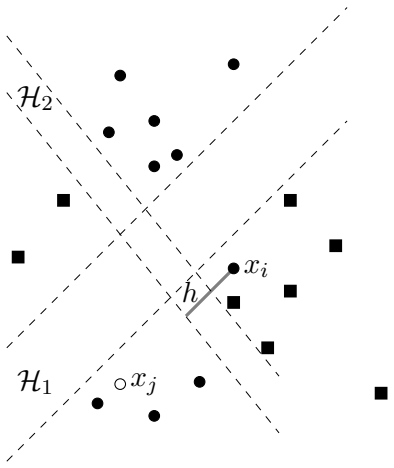


Figure 2.6: Illustration of the wrong-classification errors.

With these variables, we can model the in-margin errors by means of the following constraints:

$$\omega_r' x_i + \omega_{r0} \geq 1 - e_{ir} - T\,(3 - t_{ir} - t_{jr} - h_{ij}), \qquad \forall r = 1, \ldots, m, \tag{2.10}$$

$$\omega_r' x_i + \omega_{r0} \leq -1 + e_{ir} + T\,(1 + t_{ir} + t_{jr} - h_{ij}), \qquad \forall r = 1, \ldots, m. \tag{2.11}$$

These constraints model, by using the sign-patterns given by t, that, $e_{ir} = \max\{0, \min\{1, 1 - \mathrm{s}_r(x)(\omega_r' x_i + \omega_{r0})\}\}$. Note that the constraints are active if either $t_{ir} = t_{jr} = h_{ij} = 1$, i.e., if the well-classified observation x_j is the representative observation for x_i and both are in the positive side of the rth-hyperplane; or $t_{ir} = t_{jr} = 0$ and $h_{ij} = 1$, i.e., if the well-classified observation x_j is the representative observation for x_i and both are in the negative side of the rth-hyperplane. Thus, constraints (2.10) and (2.11) adequately model the in-margin errors for all observations . Furthermore, because of (2.3) and (2.4), and those described above, the variables e_{ir} always take values

smaller than or equal to 1.

Finally, the third addend, the out-margin errors, will be modeled through the continuous variables $d_{ir} \geq 0$, for $i = 1, \ldots, n$, $r = 1, \ldots, m$. With the set of variables described above, the out-margin misclassification errors can be adequately modeled through the following constraints:

$$d_{ir} \geq 1 - \omega_r' x_i - \omega_{r0} - T\,(2 + t_{ir} - t_{jr} - h_{ij}), \quad \forall i, j = 1, \ldots, n(y_i = y_j),\ r = 1, \ldots, m, \tag{2.12}$$

$$d_{ir} \geq 1 + \omega_r' x_i + \omega_{r0} - T\,(2 - t_{ir} + t_{jr} - h_{ij}), \quad \forall i, j = 1, \ldots, n(y_i = y_j),\ r = 1, \ldots, m. \tag{2.13}$$

Constraints (2.12) are active only if $t_{ir} = 0$ and $t_{jr} = h_{ij} = 1$, that is, if x_j is a well-classified observation in the positive side of $\mathcal{H}_r$, while x_i is wrong-classified in the negative side of $\mathcal{H}_r$ being x_j the representative observation for x_i (note that if x_i is well-classified then $h_{ii} = 1$ by (2.9) and then, the constraint cannot be activated). The second set of constraints, namely (2.13), can be analogously justified in terms of the negative side of $\mathcal{H}_r$. The main difference of these constraints with respect to (2.10) and (2.11) is that (2.12) and (2.13) are active only if x_i is wrong-classified.

According to the above constraints, a misclassified observation x_i is penalized in two ways with respect to each hyperplane $\mathcal{H}_r$. In case that x_i is well-classified with respect to $\mathcal{H}_r$, but it belongs to the margin, then $e_{ir} = 1 - \text{sign}(\omega_r' x_i + \omega_{r0})(\omega_r^t x_i + \omega_{r0}) \leq 1$ and $d_{ir} = 0$ ($t_{ir} = t_{jr}$). Otherwise, if x_i is wrong-classified with respect to $\mathcal{H}_r$, then $d_{ir} = 1 - \text{sign}(\omega_r' x_i + \omega_{r0})(\omega_r^t x_j + \omega_{r0}) \geq 1$ and $e_{ir} = 0$ ($h_{ij} = 1$ and $t_{ir} \neq t_{jr}$).

We illustrate the rationale of the proposed constraints on the data drawn in Figure 2.7. Observe that A is not correctly classified since it lies within a cell in which the blue-class is not assigned. Suppose that B, a well-classified observation, is the representative of A ($h_{AB} = 1$), then the model would have to penalize two types of errors. The first one with respect to $\mathcal{H}_2$. If we suppose $t_{B2} = 1$, then $t_{A2} = 0$, leading to an activation on constraint (2.12) being $d_{A2} > 0$. On the other hand, even though A is well-classified with respect to $\mathcal{H}_1$, we also have to penalize its margin violation. Again, if we assume $t_{B1} = 1$, then $t_{A1} = 1$, what would activate constraint (2.10) being $e_{A1} > 0$.

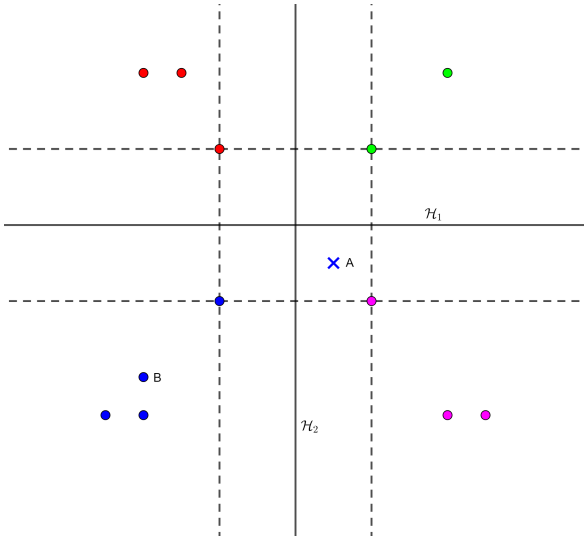


Figure 2.7: Illustration of the in-margin and out-margin constraints of our model.

The previous comments can be summarized in the following Mathematical Programming formulation for the problem:

$$\min \ \|\omega_1\|^2 + c_1 \sum_{i=1}^{n}\sum_{r=1}^{m} e_{ir} + c_2 \sum_{i=1}^{n}\sum_{r=1}^{m} d_{ir} \tag{MCSVM}$$

s.t. (2.2) – (2.13) and additionally,

$$\begin{aligned}
&\omega_r \in \mathbb{R}^p, \ \omega_{r0} \in \mathbb{R}, && \forall r = 1, \dots, m,\\
&d_{ir}, \ e_{ir} \in \mathbb{R}_+, \ t_{ir} \in \{0,1\} && \forall i = 1, \dots, n, \ r = 1, \dots, m,\\
&h_{ij} \in \{0,1\}, && \forall i, j = 1, \dots, n,\\
&z_{is} \in \{0,1\}, && \forall i = 1, \dots, n, \ s = 1, \dots, k,\\
&\xi_i \in \{0,1\}, && \forall i = 1, \dots, n.
\end{aligned}$$

(MCSVM) is a mixed integer non linear programming model, whose nonlinear terms come from the norm minimization in the objective function and constraints (2.2), so that they are second order cone representable. In case one chooses the ℓ_1-norm instead of the Euclidean norm, the model becomes a mixed integer linear programming problem. Therefore, the model is suitable to be solved using any of the available commercial solvers, as Gurobi, CPLEX, etc. The main bottleneck of the above formulation relies on the number $O(n^2)$ of binary variables.

Remark 2.1 (Ramp Loss misclassification errors)**.** *An alternative measure of misclassification training errors is the ramp loss. The ramp loss version of the model is interesting for certain instances since it allows one to improve the robustness against potential outliers. Instead of using out of margin hinge loss errors h_O, the ramp-loss*

measure consists of penalizing wrong-classified observations by a constant, independently on how far they are from being well-classified. Given an observation/label, (x, y), the ramp-loss with respect to $\mathbb{H}$, is defined as:

$$\mathrm{RL}(x, y, \mathbb{H}) = \begin{cases} 0 & \text{if } x \text{ is well-classified} \\ 1 & \text{otherwise} \end{cases}$$

Note that, for the training sample, the ramp-loss is represented in our model through the ξ-variables. More specifically, $\mathrm{RL}(x_i, y_i, \mathbb{H}) = \xi_i$ for all $i = 1, \ldots, n$. In order to do that we just need to introduce the following modifications on the MINLP problem:

$$\min \ \|\omega_1\|^2 + c_1 \sum_{i=1}^{n} \sum_{r=1}^{m} e_{ir} + c_2 \sum_{i=1}^{n} \xi_i \qquad (\mathrm{MCSVM_{RL}})$$

s.t. (2.2) − (2.11) *and additionally,*

$$\begin{aligned}
&\omega_r \in \mathbb{R}^p, \ \omega_{r0} \in \mathbb{R}, && \forall r = 1, \ldots, m, \\
&e_{ir} \in \mathbb{R}_+, \ t_{ir} \in \{0, 1\}, && \forall i = 1, \ldots, n, \ r = 1, \ldots, m, \\
&h_{ij} \in \{0, 1\}, && \forall i, j = 1, \ldots, n, \\
&z_{is} \in \{0, 1\}, && \forall i = 1, \ldots, n, \ s = 1, \ldots, k, \\
&\xi_i \in \{0, 1\}, && \forall i = 1, \ldots, n.
\end{aligned}$$

Remark 2.2 (Controlling misclassification errors while training the model)**.** *The benefits of imposing a minimum accuracy, sensitivity, or specificity on the training sample of binary SVM models has been recently analyzed by Benítez-Peña et al. (2019). As it has been pointed out before, the ξ-variables in our model allow us to know whether an observation is well or wrong-classified. Hence, using these variables one may impose a minimum desired accuracy on the training sample in the multiclass scenario. Given a minimum threshold for the accuracy, $\mu \in (0, 1]$, the following constraint enforces to construct a classification rule with at least an accuracy of μ:*

$$\frac{1}{n} \sum_{i=1}^{n} \xi_i \leq 1 - \mu.$$

Observe that the left-hand-side of the inequality indicates the proportion of wrong-classified observations. Thus, by fixing $\mu = 1$, our models guarantee a perfect classification on the training sample for a large enough number of hyperplanes (m), independently of the chosen kernel. This property is especially outstanding in the linear kernel case, in which other multiclass SVM methods may not achieve a perfect classification on some training samples.

Moreover, the same analysis can be applied to subset of classes, extending the notions of specificity and sensibility in binary classification to a multiclass classification rule. Actually, if one desires to impose a minimum true rate on a given subset of the classes, $S \subseteq \{y_1, \ldots, y_k\}$, one can easily modify the previous constraint as follows:

$$\frac{1}{|S|} \sum_{\substack{i=1 \\ y_i \in S}}^{n} \xi_i \leq 1 - \mu.$$

2.3.1 Building the classification rule

Keep in mind that the main goal of multiclass classification is to determine a decision rule such that, given any observation, it is able to assign it a class, i.e., to determine the optimal suitable assignment. Hence, once the solution of (MCSVM) is obtained, the decision rule has to be derived. Given $x \in \mathbb{R}^p$, two different situations are possible: (a) x belongs to a cell with an assigned class; and (b) x belongs to a cell with no training observations inside, so with non assigned class. For the first case, x is assigned to its cell's class. In the second case, different strategies to determine a class for x are possible.

We propose the following assignment rule based on the same allocation methods used in (MCSVM): observations are assigned to their closest well-classified representatives. More specifically, let $\mathrm{s}(x)$ be the sign-pattern of x with respect to the optimal arrangement of hyperplanes $\mathbb{H}^* = \{(\omega_1^*, \omega_{10}^*), \ldots, (\omega_m^*, \omega_{m0}^*)\}$ obtained from (MCSVM), and let $J = \{j \in \{1, \ldots, n\} : \xi_j^* = 0\}$ (here ξ^* stands for the optimal vector obtained by solving (MCSVM)). Then, among all the well-classified observations in the training sample, J, we assign to x the class of the one whose cell is the *closest* (less separated from x). Such a classification of x can be obtained by enumerating all the possible assignments, $O(|J|)$ and computing the distance measure over all of them. Equivalently, one can solve the following Mathematical Programming problem:

$$\begin{aligned} \min & \sum_{j \in J} \sum_{\substack{r=1 \\ s(x_j)_r + \mathrm{s}(x)_r = 0,}}^{m} \gamma_j \left| (\omega_r^*)' x + \omega_{r0}^* \right| \\ \text{s.t.} & \sum_{j \in J} \gamma_j = 1, \\ & \gamma_j \in \{0, 1\}, \quad \forall j \in J, \end{aligned}$$

where $\gamma_j = \begin{cases} 1 & \text{if } x \text{ is assigned to the same cell as } x_j, \\ 0 & \text{otherwise.} \end{cases}$

The integrality condition in the problem above can be relaxed, since the unique constraint in the problem is totally unimodular and thus, the problem is a linear programming problem. Clearly, the solution of the above problem gives the optimal labelling of x with respect to the existing cells in the arrangement.

One could also consider other robust measures for such an assignment following the same paradigm, as min-max error or the like.

2.3.2 Non-Linear classifiers

Finally, we analyze a crucial question in any SVM-based methodology, which is whether one can apply the Theory of Kernels in our framework. Using kernels means being able to map the observations (via some transformation $\varphi : \mathbb{R}^p \to \mathbb{R}^P$) to a higher dimensional space, where data separation is more adequately performed. If the desired transformation, φ, is known, one could transform the data and solve the problem (MCSVM) with a higher number of variables. However, in binary SVM, when formulating the dual of the classification problem, one can observe that it only depends on the original data via the inner products of each pair of observations (originally in $\mathbb{R}^p$), i.e., through the amounts $x_i' x_j$ for $i, j = 1, \ldots, n$. If the transformation φ is applied to the data, the observations only appear in the (classical SVM) problem as $\varphi(x_i)'\varphi(x_j)$ for $j = 1, \ldots, n$. Thus, kernels are defined as generalized inner products as $K(a, b) = \varphi(a)'\varphi(b)$ for each $a, b \in \mathbb{R}^p$, and they can be introduced using any of the well known families of kernel functions (see e.g., Horn et al. (2018)). Moreover, Mercer's theorem gives sufficient conditions for a function $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ to be a kernel function (one which is constructed as the inner product of a transformation of the features). This result allows one to construct kernel measures that induce transformations. The main advantage of using kernels, apart from a probably better separation in the projected space, is that in binary SVM, the complexity of the *transformed problem is the same as the original one.* More specifically, the dual problems have the same structure and the same number of variables.

Although problem (MCSVM) is a MINLP, and then, duality results do not hold, one can apply decomposition techniques to separate the binary and the continuous variables and then, iterate over the binary variables by recursively solving certain continuous and easier problems (see e.g., Benders (1962); Geoffrion (1972)). The following result, whose proof can be found in the Apendix, states that our approach also allows us to find nonlinear classifiers via the kernel tools.

This result is interesting by itself since links the general theory of nonlinear classifiers, very well known for the standard SVM theory with Euclidean distance, to our multiclass framework. It is worth noting that for a function $h_H(\mathcal{H}_1, \ldots, \mathcal{H}_m) = \sum_{r=1}^{m} \|\omega_r\|^2$ the usual kernel trick construction applies *mutatis-mutandis*. Never-

theless, as already pointed out, we elaborate our approach based on the natural measure of margin that maximizes the minimum separation between classes, namely $h_H(\mathcal{H}_1,\ldots,\mathcal{H}_m) = \max\{\|\omega_1\|^2,\ldots,\|\omega_m\|^2\}$. This change implies that the mathematical development known for the standard kernel trick does not carry over our new approach without a further analysis. We prove below that in this new framework one can also find nonlinear multiclass classifiers that, as in the standard SVM case, only depend on the transformation by means of inner products of the original data. Hence, extending the kernel trick to this multiclass framework.

Theorem 2.1. *Let $\varphi : \mathbb{R}^p \to \mathbb{R}^P$ be a transformation of the feature space. Then, one can obtain a multiclass classifier which only depends on the original data by means of the inner products $\varphi(x_i)'\varphi(x_j)$, for $i,j = 1,\ldots,n$.*

Proof. Note that once the binary variables of our model are fixed, the problem becomes polynomial time solvable and it reduces to find the coordinates of the coefficients and intercepts of the hyperplanes and the different misclassifying errors. In particular, it is clear that the MINLP formulation for the problem is equivalent to:

$$
\begin{aligned}
&\min_{h,z,t,\xi} \ \Phi(h,z,t,\xi)\\
&\text{s.t. } (2.5)-(2.9),\\
&\qquad h_{ij} \in \{0,1\}, && \forall i,j = 1,\ldots,n,\\
&\qquad t_{ir} \in \{0,1\}, && \forall i = 1,\ldots,n,\ r = 1,\ldots,m,\\
&\qquad z_{is} \in \{0,1\}, && \forall i = 1,\ldots,n,\ s = 1,\ldots,k,\\
&\qquad \xi_i \in \{0,1\}, && \forall i = 1,\ldots,n.
\end{aligned}
$$

where Φ is the evaluation of the margin and hinge-loss errors for any assignment provided by the binary variables. That is,

$$
\begin{aligned}
\Phi(h,z,t,\xi) = \min_{\omega,\omega_0,e,d} \ & \frac{1}{2}\|\omega_1\|^2 + c_1\sum_{i=1}^{n}\sum_{r=1}^{m} e_{ir} + c_2\sum_{i=1}^{n}\sum_{r=1}^{m} d_{ir} && (\phi_{val})\\
&\text{s.t. } (2.2),(2.10)-(2.13),\\
&\omega_r \in \mathbb{R}^p, \omega_{r0} \in \mathbb{R}, && \forall r = 1,\ldots,m,\\
&d_{ir}, e_{ir} \geq 0, && \forall i = 1,\ldots,n,\ r = 1,\ldots,m.
\end{aligned}
$$

The above problem would be separable provided that the first $m-1$ constraints (2.2) were relaxed. For the sake of simplicity in the notation, we consider the

following functions, $\kappa^\ell : \{0,1\} \to \mathbb{R}$ for $\ell = 1,2,3$, defined as:

$$\kappa^1(t) := T(1-t), \;\; \kappa^2(t) := Tt, \;\; \kappa^3(t) := -1 + Tt$$

for $t \in \{0,1\}$. Note that $\kappa^1(0) = \kappa^2(1) = T$, $\kappa^1(1) = \kappa^2(0) = 0$, and that $\kappa^3(0) = -1$, $\kappa^3(1) = T - 1$.

Based on the separability mentioned above, we introduce another instrumental family of problems for all $r = 2, \ldots, m$, namely,

$$\begin{aligned}
\Phi_r(h,z,t,\xi,\omega_1) \;\; = \;\; \min_{\omega_r,\omega_{r0},e,d} \; & c_1 \sum_{i=1}^{n} e_{ir} + c_2 \sum_{i=1}^{n} d_{ir} \\
\text{s.t.} \;\; & \frac{1}{2}\|\omega_r\|^2 - \frac{1}{2}\|\omega_1\|^2 \leq 0, \\
& -\omega_{ir}' x_i - \omega_{r0} \leq \kappa^1(t_{ir}), \forall i = 1, \ldots, n, \\
& \omega_{ir}' x_i + \omega_{r0} \leq \kappa^2(t_{ir}), \forall i = 1, \ldots, n, \\
& -\omega_r' x_i - \omega_{r0} - e_{ir} \leq \kappa^3(u_{ijr}^+), \forall i,j = 1, \ldots, n, \\
& \omega_r' x_i + \omega_{r0} - e_{ir} \leq \kappa^3(u_{ijr}^-), \forall i,j = 1, \ldots, n, \\
& -\omega_r' x_i - \omega_{r0} - d_{ir} \leq \kappa^3(q_{ijr}^+), \forall i,j = 1, \ldots, n \; (y_i = y_j), \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\mathrm{SP}_r) \\
& \omega_r' x_i + \omega_{r0} - d_{ir} \leq \kappa^3(q_{ijr}^-), \forall i,j = 1, \ldots, n \; (y_i = y_j), \\
& -d_{ir} \leq 0, \forall i = 1, \ldots, n, \\
& -e_{ir} \leq 0, \forall i = 1, \ldots, n, \\
& \omega_r \in \mathbb{R}^p, \omega_{r0} \in \mathbb{R},
\end{aligned}$$

where for simplifying the notation we have introduced the auxiliary variables $u_{ijr}^+ := 3 - t_{ir} - t_{jr} - h_{ij}$, $u_{ijr}^- := 1 + t_{ir} + t_{jr} - h_{ij}$, $q_{ijr}^+ = 3 + t_{ir} - h_{ij} - t_{jr} - \xi_i$ and $q_{ijr}^- = 3 + t_{jr} - h_{ij} - t_{ir} - \xi_i$, for $i,j = 1, \ldots, n$, and $r = 2, \ldots, m$.

Observe that Φ_r, apart from the first constraint, only considers variables associated to the rth hyperplane.

Moreover, we need another problem that accounts for the first part of Φ.

$$
\begin{aligned}
\Phi_1(h,z,t,\xi) = \min_{\omega_1,\omega_{10},e,d} \; & \frac{1}{2}\|\omega_1\|^2 + c_1\sum_{i=1}^{n} e_{i1} + c_2\sum_{i=1}^{n} d_{i1} \\
\text{s.t.} \; & -\omega_1' x_i - \omega_{10} \leq \kappa^1(t_{i1}), \forall i = 1,\ldots,n, \\
& \omega_1' x_i + \omega_{10} \leq \kappa^2(t_{i1}), \forall i = 1,\ldots,n, \\
& -\omega_1' x_i - \omega_{10} - e_{i1} \leq \kappa^3(u^+_{ij1}), \forall i,j = 1,\ldots,n, \\
& \omega_1' x_i + \omega_{10} - e_{i1} \leq \kappa^3(u^-_{ij1}), \forall i,j = 1,\ldots,n, \\
& -\omega_1' x_i - \omega_{10} - d_{i1} \leq \kappa^3(q^+_{ij1}), \forall i,j = 1,\ldots,n \; (y_i = y_j), \qquad (\mathrm{SP}_1) \\
& \omega_1' x_i + \omega_{10} - d_{i1} \leq \kappa^3(q^-_{ij1}), \forall i,j = 1,\ldots,n \; (y_i = y_j), \\
& -d_{i1} \leq 0, \forall i = 1,\ldots,n, \\
& -e_{i1} \leq 0, \forall i = 1,\ldots,n, \\
& \omega_1 \in \mathbb{R}^p, \omega_{10} \in \mathbb{R}.
\end{aligned}
$$

Thus, using the above notation, (MCSVM) is equivalent to the following problem:

$$
\begin{aligned}
\min_{h,z,t,\xi,\omega_1} \; & \Phi_1(h,z,t,\xi) + \sum_{r=2}^{m} \Phi_r(h,z,t,\xi,\omega_1) \\
\text{s.t.} \; & (2.5) - (2.9) \\
& h_{ij} \in \{0,1\}, && \forall i,j = 1,\ldots,n, \\
& t_{ir} \in \{0,1\}, && \forall i = 1,\ldots,n, \; r = 1,\ldots,m, \\
& z_{is} \in \{0,1\}, && \forall i = 1,\ldots,n, \; s = 1,\ldots,k, \\
& \xi_i \in \{0,1\}, && \forall = 1,\ldots,n.
\end{aligned}
$$

Observe that the above problem only accounts for ω_1 and the binary variables. Once they are fixed can be plugged into the SP_r, $r = 1,\ldots,m$ subproblems and then it allows one to find the optimal values of the continuous variables. The elements $\kappa^1(t_{ir}), \kappa^2(t_{ir}), \kappa^3(u^+_{ijr}), \kappa^3(u^-_{ijr}), \kappa^3(q^+_{ijr})$, and $\kappa^3(q^-_{ijr})$ are fixed constants once the binary variables are fixed.

In order to solve the problem for a fixed set of binary variables we can proceed recursively, solving independently (SP_r) for all $r = 2,\ldots,m$, and then combining their solutions with (SP_1).

Therefore, to get that goal we apply Lagrangean duality to obtain and exact dual reformulation. Indeed, relaxing the constraints of (SP_r) with dual multipliers $\mu_r \geq 0$ (assuming that $\mu^0_r \neq 0$) and denoting by $\alpha_{ir} = \alpha_{ir} = \mu^1_{ir} - \mu^2_{ir} + \sum_{j:y_i=y_j} (\mu^3_{ijr} - \mu^4_{ijr} + \mu^5_{ijr} - \mu^6_{ijr})$, for all $i = 1,\ldots,n$ and $r = 1,\ldots,m$; after some derivations that

can be followed in Lemma 2.1, one can check that evaluating Φ for given values of the binary variables, can be obtained solving the continuous optimization problem (JSP). (All details can be found in Lemma 2.1.)

Next, we analyze how the evaluation of the function ϕ, given in (ϕ_{val}), depends on the original data. In order to do that we find the optimal solutions of (JSP). Dualizing the constraints that correspond to Φ_1 with multipliers $\mu_1 \geq 0$ and those where the variables Γ_r appear, with dual multipliers γ_r, the Lagrangean function of the problem (JSP) results in:

$$\begin{aligned}
\mathcal{L}(\omega_1, \omega_{01}, d, e; \mu) = \frac{1}{2}\|\omega_1\|^2(1 - \sum_{r=2}^{m} \gamma_r \mu_r^0) + c_1 \sum_{i=1}^{n} e_{i1} + c_2 \sum_{i=1}^{n} d_{i1} + \sum_{r=2}^{m} \Gamma_r(1 - \gamma_r) \\
+ \sum_{r=2}^{m} \left(\frac{\gamma_r}{2\mu_r^0} \sum_{i,j=1}^{n} \alpha_{ir}\alpha_{jr} x_i' x_j + \sum_{i,j:u_{ijr}^+=0} \mu_{ijr}^3 \right. \\
\left. + \sum_{i,j:u_{ijr}^-=0} \mu_{ijr}^4 + \sum_{i,j:q_{ijr}^+=0} \mu_{ijr}^5 + \sum_{i,j:q_{ijr}^-=0} \mu_{ijr}^6 \right) \\
+ \sum_{i=1}^{n} \mu_{i1}^1 \Big(- \omega_1' x_i - \omega_{10} - \kappa^1(t_{i1}) \Big) \\
+ \sum_{i=1}^{n} \mu_{i1}^2 \Big(\omega_1' x_i + \omega_{10} - \kappa^2(t_{i1}) \Big) \\
+ \sum_{i=1}^{n} \mu_{ij1}^3 \Big(- \omega_1' x_i - \omega_{10} - e_{i1} - \kappa^3(\bar{u}_{ij1}^+) \Big) \\
+ \sum_{i=1}^{n} \mu_{ij1}^4 \Big(\omega_1' x_i + \omega_{10} - e_{i1} - \kappa^3(\bar{u}_{ij1}^-) \Big) \\
+ \sum_{i,j=1}^{n} \mu_{ij1}^5 \Big(- d_{ir} - \omega_1' x_i - \omega_{10} - \kappa^3(\bar{q}_{ij1}^+) \Big) \\
+ \sum_{i,j=1}^{n} \mu_{ij1}^6 \Big(- d_{ir} + \omega_1' x_i + \omega_{10} - \kappa^3(\bar{q}_{ij1}^-) \Big) \\
+ \sum_{i=1}^{n} \mu_{i1}^7(-d_{i1}) + \sum_{i=1}^{n} \mu_{i1}^8(-e_{i1}),
\end{aligned}$$

and its KKT optimality conditions reduce to:

- $(1 - \sum_{r=2}^{m} \gamma_r \mu_r^0)\omega_1 = \sum_{i=1}^{n} \Big(\mu_{i1}^1 - \mu_{i1}^2 + \sum_{j:y_i=y_j} (\mu_{ij1}^3 - \mu_{ij1}^4 + \mu_{ij1}^5 - \mu_{ij1}^6) \Big) x_i.$

- $\sum_{i=1}^{n} \Big(\mu_{i1}^1 - \mu_{i1}^2 + \sum_{j:y_i=y_j} (\mu_{ij1}^3 - \mu_{ij1}^4 + \mu_{ij1}^5 - \mu_{ij1}^6) \Big) = 0.$
- $\sum_{j:y_i=y_j} (\mu_{ij1}^3 + \mu_{ij1}^4) + \mu_{i1}^8 = c_1, \forall i = 1, \ldots, n.$
- $\sum_{j:y_i=y_j} (\mu_{ijr}^5 + \mu_{ijr}^6) + \mu_{ir}^7 = c_2, \forall i = 1, \ldots, n.$
- $1 - \gamma_r = 0, \forall r = 2, \ldots, m.$
- $\mu \geq 0, \gamma \geq 0.$

Using the same notation as before, $\alpha_{i1} = \mu_{i1}^1 - \mu_{i1}^2 + \sum_{j:y_i=y_j} (\mu_{ij1}^3 - \mu_{ij1}^4 + \mu_{ij1}^5 - \mu_{ij1}^6)$, for all $i = 1, \ldots, n$, and simplifying the expressions using the KKT conditions and the complementary slackness conditions we get that the strong dual of (JSP) is:

$$
\begin{aligned}
\max_{\omega,\omega_0,d,e;\mu} \; & \frac{1}{2(1-\sum_{r=2}^{m}\mu_r^0)} \sum_{i,j=1}^{n} \alpha_{i1}\alpha_{j1} x_i' x_j + \sum_{i,j:u_{ij1}^{+}=0} \mu_{ijr}^{3} + \sum_{i,j:u_{ij1}^{-}=0} \mu_{ij1}^{4} \\
& + \sum_{i,j:q_{ij1}^{+}=0} \mu_{ijr}^{5} + \sum_{i,j:q_{ij1}^{-}=0} \mu_{ij1}^{6} + \sum_{r=2}^{m} \left(\frac{1}{2\mu_r^0} \sum_{i,j=1}^{n} \alpha_{ir}\alpha_{jr} x_i' x_j \right. \\
& \left. + \sum_{i,j:u_{ijr}^{+}=0} \mu_{ijr}^{3} + \sum_{i,j:u_{ijr}^{-}=0} \mu_{ijr}^{4} + \sum_{i,j:q_{ijr}^{+}=0} \mu_{ijr}^{5} + \sum_{i,j:q_{ijr}^{-}=0} \mu_{ijr}^{6} \right) \\
\text{s.t.} \; & \sum_{i=1}^{n} \alpha_{ir} = 0, \forall r = 1,\ldots,m, \qquad \text{(DJSP)} \\
& \sum_{i=1}^{n} \alpha_{i1} x_i = (1-\sum_{r=2}^{m}\mu_r^0)\omega_1, \\
& \sum_{i=1}^{n} \alpha_{ir} x_i = \mu_r^0 \omega_r, \forall r \geq 2, \\
& \sum_{j:u_{ijr}^{+}=0} \mu_{ijr}^{3} + \sum_{j:u_{ijr}^{-}=0} \mu_{ijr}^{4} \leq c_1, \forall i = 1,\ldots,n, \\
& \sum_{j:q_{ijr}^{+}=0} \mu_{ijr}^{5} + \sum_{j:q_{ijr}^{-}=0} \mu_{ijr}^{6} \leq c_2, \forall i = 1,\ldots,n, \; r = 1,\ldots,m, \\
& \mu_{ir}^{1} = 0, \forall i = 1,\ldots,n, \; r = 1,\ldots,m, \text{ with } \bar{t}_{ir} = 0, \\
& \mu_{ir}^{2} = 0, \forall i = 1,\ldots,n, \; r = 1,\ldots,m, \text{ with } \bar{t}_{ir} = 1, \\
& \alpha_{ir} = \mu_{ir}^{1} - \mu_{ir}^{2} + \sum_{i,j:u_{ij1}^{+}=0} \mu_{ijr}^{3} - \sum_{i,j:u_{ij1}^{-}=0} \mu_{ijr}^{4} + \sum_{i,j:q^{-}+ij1=0} \mu_{ijr}^{5} - \sum_{i,j:q_{ij1}^{-}=0} \mu_{ijr}^{6}, \\
& \forall i = 1,\ldots,n, \\
& \mu_{\mathbf{r}} \geq 0, \in \mathbb{R}, \; \forall r = 1,\ldots,m.
\end{aligned}
$$

Note that the objective function of (DJSP) only depends of the x-input data through the inner products $x_i' x_j$ for $i,j = 1,\ldots,n$, and also the expressions of ω_1 from the dual variables is given as:

$$(1-\sum_{r=2}^{m}\mu_r^0)\omega_1 = \sum_{i=1}^{n} \alpha_{i1} x_i.$$

The dependence of ω_1 with other ω_r in the primal formulation is only through the nonincreasing sorted values of $\|\omega_1\|,\ldots,\|\omega_m\|$ in which we now that the largest value is $\|\omega_1\|$. Thus, solving the dual problem (DJSP) allows us to determine all the optimal hyperplanes ω_r, for all $r = 1,\ldots,m$. In case a transformation φ is

performed to the input data, the dependence of the data in the problem will be through the inner products $\varphi(x_i)'\varphi(x_j)$ for all $i, j = 1, \ldots, n$, and the kernel Theory can be applied.

□

Lemma 2.1. *The evaluation of Φ for given values of the binary variables $\bar{t}$, $\bar{\xi}$, $\bar{h}$ and $\bar{z}$ can be obtained solving the following continuous optimization problem:*

$$
\begin{aligned}
\hat{\Phi}(h,z,t,\xi) = \min & \left\{ \frac{1}{2}\|\omega_1\|^2 + c_1 \sum_{i=1}^{n} e_{i1} + c_2 \sum_{i=1}^{n} d_{i1} + \sum_{r=2}^{m} \Gamma_r \right\} \\
& s.t. \quad \left(-\frac{\mu_r^0}{2}\|\omega_1\|^2 + \frac{1}{2\mu_r^0} \sum_{i,j=1}^{n} \alpha_{ir}\alpha_{jr} x_i' x_j \right. \\
& + \sum_{i,j:u_{ijr}^+=0} \mu_{ijr}^3 + \sum_{i,j:u_{ijr}^-=0} \mu_{ijr}^4 \\
& \left. + \sum_{i,j:q_{ijr}^+=0} \mu_{ijr}^5 + \sum_{i,j:q_{ijr}^-=0} \mu_{ijr}^6 + c_1 \right) \leq \Gamma_r, \forall r \geq 2, \\
& -\omega_1' x_i - \omega_{10} \leq \kappa^1(t_{i1}), \forall i = 1, \ldots, n, \\
& \omega_1' x_i + \omega_{10} \leq \kappa^2(t_{i1}), \forall i = 1, \ldots, n, \\
& -\omega_1' x_i - \omega_{10} - e_{i1} \leq \kappa^3(u_{ij1}^+), \forall i, j = 1, \ldots, n, \\
& \omega_1' x_i + \omega_{10} - e_{i1} \leq \kappa^3(u_{ij1}^-), \forall i, j = 1, \ldots, n, \\
& -\omega_1' x_i - \omega_{10} - d_{i1} \leq \kappa^3(q_{ij1}^+), \forall i, j = 1, \ldots, n, \ (y_i = y_j), \qquad \text{(JSP)} \\
& \omega_1' x_i + \omega_{10} - d_{i1} \leq \kappa^3(q_{ij1}^-), \forall i, j = 1, \ldots, n, \ (y_i = y_j), \\
& -d_{i1} \leq 0, i = 1, \ldots, n, \\
& -e_{i1} \leq 0, i = 1, \ldots, n, \\
& \omega_1 \in \mathbb{R}^p, \omega_{10} \in \mathbb{R}, \\
& \sum_{j:u_{ijr}^+=0} \mu_{ijr}^3 + \sum_{j:u_{ijr}^-=0} \mu_{ijr}^4 + \mu_{ir}^8 \geq c_1, i = 1, \ldots, n, \\
& \sum_{j:q_{ijr}^+=0} \mu_{ijr}^5 + \sum_{j:q_{ijr}^-=0} \mu_{ijr}^6 \leq c_2, \forall i = 1, \ldots, n, \ r = 1, \ldots, m, \\
& \mu_{ir}^1 = 0, \forall i = 1, \ldots, n, \ r = 1, \ldots, m, \ \textit{with } \bar{t}_{ir} = 0, \\
& \mu_{ir}^2 = 0, \forall i = 1, \ldots, n, \ r = 1, \ldots, m, \ \textit{with } \bar{t}_{ir} = 1, \\
& \alpha_{ir} = \mu_{ir}^1 - \mu_{ir}^2 + \sum_{i,j:u_{ij1}^+=0} \mu_{ijr}^3 - \sum_{i,j:u_{ij1}^-=0} \mu_{ijr}^4 + \sum_{i,j:q_{ij1}^+=0} \mu_{ijr}^5 - \sum_{i,j:q_{ij1}^-=0} \mu_{ijr}^6, \\
& \forall i = 1, \ldots, n, \\
& \mu \geq 0.
\end{aligned}
$$

where $\mu_r \geq 0$ are the dual multipliers of the constraints in Problem (SP_r).

Proof. In order to prove the result, in what follows we derive the optimality conditions of the Lagrangean dual of (SP_r) for $r = 2, \ldots, m$. Relaxing the constraints of (SP_r) with dual multipliers $\mu_r \geq 0$, the Lagrangean function for given values of the binary variables $\bar{t}$, $\bar{\xi}$, $\bar{h}$ and $\bar{z}$ (and consequently for values $\bar{u}^+$, $\bar{u}^-$, $\bar{q}^+$ and $\bar{q}^-$) is:

$$\begin{aligned}
\mathcal{L}_r(\omega_r, \omega_{0r}, d, e; \mu) = c_1 \sum_{i=1}^{n} e_{ir} + c_2 \sum_{i=1}^{n} d_{ir} + \mu_r^0 \Big(\frac{1}{2}\|\omega_r\|^2 - \frac{1}{2}\|\omega_1\|^2 \Big) \\
+ \sum_{i=1}^{n} \mu_{ir}^1 \Big(-\omega_r' x_i - \omega_{r0} - \kappa^1(t_{ir}) \Big) + \sum_{i=1}^{n} \mu_{ir}^2 \Big(\omega_r' x_i + \omega_{r0} - \kappa^2(t_{ir}) \Big) \\
+ \sum_{i,j=1}^{n} \mu_{ijr}^3 \Big(-\omega_r' x_i - \omega_{r0} - e_{ir} - \kappa^3(\bar{u}_{ijr}^+) \Big) + \sum_{i,j=1}^{n} \mu_{ijr}^4 \Big(\omega_r' x_i + \omega_{r0} - e_{ir} - \kappa^3(\bar{u}_{ijr}^-) \Big) \\
+ \sum_{i,j=1}^{n} \mu_{ijr}^5 \Big(-d_{ir} - \omega_r' x_i - \omega_{r0} - \kappa^3(\bar{q}_{ijr}^+) \Big) + \sum_{i,j=1}^{n} \mu_{ijr}^6 \Big(-d_{ir} + \omega_r' x_i + \omega_{r0} - \kappa^4(\bar{q}_{ijr}^-) \Big) \\
+ \sum_{i=1}^{n} \mu_{ir}^7 (-d_{ir}) + \sum_{i=1}^{n} \mu_{ir}^8 (-e_{ir}).
\end{aligned}$$

Therefore, the KKT optimality conditions read as:

- $\mu_r^0 \omega_r = \sum_{i=1}^{n} \Big(\mu_{ir}^1 - \mu_{ir}^2 + \sum_{j:y_i=y_j} (\mu_{ijr}^3 - \mu_{ijr}^4 + \mu_{ijr}^5 - \mu_{ijr}^6) \Big) x_i.$
- $\sum_{i=1}^{n} \Big(\mu_{ir}^1 - \mu_{ir}^2 + \sum_{j:y_i=y_j} (\mu_{ijr}^3 - \mu_{ijr}^4 + \mu_{ijr}^5 - \mu_{ijr}^6) \Big) = 0.$
- $\sum_{j:y_i=y_j} (\mu_{ijr}^3 + \mu_{ijr}^4) + \mu_{ir}^8 = c_1, \; \forall i = 1, \ldots, n.$
- $\sum_{j:y_i=y_j} (\mu_{ijr}^5 + \mu_{ijr}^6) + \mu_{ir}^7 = c_2, \; \forall i = 1, \ldots, n.$
- $\mu_r \geq 0.$

First of all, we observe that if $\|\omega_r\| < \|\omega_1\|$ at optimality then $\mu_r^0 = 0$ and actually, we do not have to consider the corresponding constraint nor the addend $\frac{\mu_r^0}{2}(\|\omega_r\|^2 - \|\omega_1\|^2)$ in the Lagrangean function. Hence, denoting by $\alpha_{ir} = \mu_{ir}^1 - \mu_{ir}^2 + \sum_{j:y_i=y_j} (\mu_{ijr}^3 - \mu_{ijr}^4 + \mu_{ijr}^5 - \mu_{ijr}^6)$, for all $i = 1, \ldots, n$, and assuming that $\mu_r^0 \neq 0$, the

dual of (SP_r) reads as:

$$\begin{aligned}
\max_{\omega_r,\omega_{r0},d,e;\mu} \; & -\frac{\mu_r^0}{2}\|\omega_1\|^2 + \frac{1}{2\mu_r^0}\sum_{i,j=1}^{n}\alpha_{ir}\alpha_{jr}x_i'x_j - \sum_{i=1}^{n}\mu_{ir}^1\kappa^1(\bar{t}_{ir}) - \sum_{i=1}^{n}\mu_{ir}^2\kappa^2(\bar{t}_{ir}) - \sum_{i=1}^{n}\mu_{ir}^3\kappa^3(\bar{u}_{ijr}^+) \\
& - \sum_{i=1}^{n}\mu_{ir}^4\kappa^4(\bar{u}_{ijr}^-) - \sum_{i,j=1}^{n}\mu_{ijr}^5\kappa^5(\bar{q}_{ijr}^+) - \sum_{i,j=1}^{n}\mu_{ijr}^6\kappa^6(\bar{q}_{ijr}^-) \\
\text{s.t.} \; & \sum_{i=1}^{n}\alpha_{ir}x_i = \mu_r^0\omega_r, \\
& \sum_{i=1}^{n}\alpha_{ir} = 0, \\
& \sum_{j:y_i=y_j}(\mu_{ijr}^3 + \mu_{ijr}^4) \le c_1, \forall i = 1,\dots,n, \\
& \sum_{j:y_i=y_j}(\mu_{ijr}^5 + \mu_{ijr}^6) \le c_2, \forall i = 1,\dots,n, \\
& \alpha_{ir} = \mu_{ir}^1 - \mu_{ir}^2 + \sum_{i,j:u_{ij1}^+=0}\mu_{ijr}^3 - \sum_{i,j:u_{ij1}^-=0}\mu_{ijr}^4 + \sum_{i,j:q_{ij1}^+=0}\mu_{ijr}^5 - \sum_{i,j:q_{ij1}^-=0}\mu_{ijr}^6, \\
& \forall i = 1,\dots,n, \\
& \mu_{\mathbf{r}} \ge 0.
\end{aligned}$$

Let us simplify further the expressions above. We observe that:

$$\sum_i \mu_{ir}^1\kappa^1(\bar{t}_{ir}) + \sum_i \mu_{ir}^2\kappa^2(\bar{t}_{ir}) = 0,$$

$$-\sum_{i,j=1}^{n}\mu_{ijr}^3\kappa^3(\bar{u}_{ijr}^+) - \sum_{i,j=1}^{n}\mu_{ijr}^4\kappa^3(\bar{u}_{ijr}^-) = \sum_{\substack{i,j:\\ \bar{u}_{ijr}^+=0}}\mu_{ijr}^3 + \sum_{\substack{i,j:\\ \bar{u}_{ijr}^-=0}}\mu_{ijr}^4,$$

and

$$-\sum_{i,j}\mu_{ijr}^5\kappa^3(\bar{q}_{ijr}^+) - \sum_{i,j}\mu_{ijr}^6\kappa^4(\bar{q}_{ijr}^-) = \sum_{\substack{i,j:\\ \bar{q}_{ijr}^+=0}}\mu_{ijr}^5 + \sum_{\substack{i,j:\\ \bar{q}_{ijr}^-=0}}\mu_{ijr}^6.$$

Using those equations and substituting the problem becomes:

$$
\begin{aligned}
\max_{\omega_r,\omega_0,d,e;\mu} \; & -\frac{\mu_r^0}{2}\|\omega_1\|^2 + \frac{1}{2\mu_r^0}\sum_{i,j=1}^{n}\alpha_{ir}\alpha_{jr}x_i'x_j + \sum_{i,j:u_{ijr}^+=0}\mu_{ijr}^3 + \sum_{i,j:u_{ijr}^-=0}\mu_{ijr}^4 \\
& + \sum_{i,j:q_{ijr}^+=0}\mu_{ijr}^5 + \sum_{i,j:q_{ijr}^-=0}\mu_{ijr}^6 \\
\text{s.t.} \; & \sum_{i=1}^{n}\alpha_{ir}x_i = \mu_r^0\omega_r, \\
& \sum_{i=1}^{n}\alpha_{ir} = 0, \forall r = 1,\dots,m, \\
& \sum_{j:u_{ijr}^+=0}\mu_{ijr}^3 + \sum_{j:u_{ijr}^-=0}\mu_{ijr}^4 \le c_1, \forall i = 1,\dots,n, \\
& \sum_{j:q_{ijr}^+=0}\mu_{ijr}^5 + \sum_{j:q_{ijr}^-=0}\mu_{ijr}^6 \le c_2, \forall i = 1,\dots,n,\ r = 1,\dots,m, \qquad (\mathrm{DSP}_r) \\
& \mu_{ir}^1 = 0, \forall i = 1,\dots,n,\ r = 1,\dots,m \text{ with } \bar{t}_{ir} = 0, \\
& \mu_{ir}^2 = 0, \forall i = 1,\dots,n,\ r = 1,\dots,m \text{ with } \bar{t}_{ir} = 1, \\
& \alpha_{ir} = \mu_{ir}^1 - \mu_{ir}^2 + \sum_{i,j:u_{ij1}^+=0}\mu_{ijr}^3 - \sum_{i,j:u_{ij1}^-=0}\mu_{ijr}^4 + \sum_{i,j:q_{ij1}^+=0}\mu_{ijr}^5 - \sum_{i,j:q_{ij1}^-=0}\mu_{ijr}^6, \\
& \forall i = 1,\dots,n, \\
& \mu \ge 0.
\end{aligned}
$$

Using the strong duality in all the subproblems (SP_r) for $r = 2,\dots,m$, we can obtain the following expansion of the join subproblem (SP_r) that allows one the evaluation of ϕ defined by ϕ_{val}.

$$
\begin{aligned}
\hat{\Phi}(h,z,t,\xi) = \min \Bigg\{ & \frac{1}{2}\|\omega_1\|^2 + c_1 \sum_{i=1}^{n} e_{i1} + c_2 \sum_{i=1}^{n} d_{i1}) \\
& + \max_{\omega_r,\omega_0,d,e;\mu} \sum_{r=2}^{m} \Bigg(-\frac{\mu_r^0}{2}\|\omega_1\|^2 + \frac{1}{2\mu_r^0} \sum_{i,j=1}^{n} \alpha_{ir}\alpha_{jr} x_i' x_j + \sum_{i,j:u_{ijr}^+=0} \mu_{ijr}^3 + \sum_{i,j:u_{ijr}^-=0} \mu_{ijr}^4 \\
& + \sum_{i,j:q_{ijr}^+=0} \mu_{ijr}^5 + \sum_{i,j:q_{ijr}^-=0} \mu_{ijr}^6 \Bigg) \\
& \text{s.t. } -\omega_1^t x_i - \omega_{10} \le \kappa^1(t_{i1}), \forall i = 1,\dots,n, \\
& \omega_1' x_i + \omega_{10} \le \kappa^2(t_{i1}), \forall i = 1,\dots,n, \\
& -\omega_1' x_i - \omega_{10} - e_{i1} \le \kappa^3(u_{ij1}^+), \forall i,j = 1,\dots,n, \\
& \omega_1' x_i + \omega_{10} - e_{i1} \le \kappa^3(u_{ij1}^-), \forall i,j = 1,\dots,n, \\
& -\omega_1' x_i - \omega_{10} - d_{i1} \le \kappa^3(q_{ij1}^+), \forall i,j = 1,\dots,n \ (y_i = y_j), \\
& \omega_1' x_i + \omega_{10} - d_{i1} \le \kappa^3(q_{ij1}^-), \forall i,j = 1,\dots,n \ (y_i = y_j), \\
& -d_{i1} \le 0, \forall i = 1,\dots,n, \\
& -e_{i1} \le 0, \forall i = 1,\dots,n, \\
& \omega_1 \in \mathbb{R}^p, \omega_{10} \in \mathbb{R}, \\
& \sum_{j:u_{ijr}^+=0} \mu_{ijr}^3 + \sum_{j:u_{ijr}^-=0} \mu_{ijr}^4 \le c_1, \forall i = 1,\dots,n, \\
& \sum_{j:q_{ijr}^+=0} \mu_{ijr}^5 + \sum_{j:q_{ijr}^-=0} \mu_{ijr}^6 \le c_2, \forall i = 1,\dots,n, \ r = 1,\dots,m \\
& \mu_{ir}^1 = 0, \forall i = 1,\dots,1, \ r = 1,\dots,m, \text{ with } \bar{t}_{ir} = 0, \\
& \mu_{ir}^2 = 0, \forall i = 1,\dots,1, \ r = 1,\dots,m, \text{ with } \bar{t}_{ir} = 1, \\
& \alpha_{ir} = \mu_{ir}^1 - \mu_{ir}^2 + \sum_{i,j:u_{ij1}^+=0} \mu_{ijr}^3 - \sum_{i,j:u_{ij1}^-=0} \mu_{ijr}^4 + \sum_{i,j:q_{ij1}^+=0} \mu_{ijr}^5 - \sum_{i,j:q_{ij1}^-=0} \mu_{ijr}^6, \\
& \forall i = 1,\dots,n, \\
& \mu \ge 0.
\end{aligned}
$$

The usual transformation of the maximum in the objective function gives rise to the equivalent reformulation of the above problem as (JSP). □

2.4 Math-heuristic approach

As mentioned in previous section, the computational burden for solving (MCSVM), that is a mixed integer non linear programming problem (in which the nonlinearities

come from the norm minimization in the objective function), is the combination of the discrete aspects and the non-linearities in the model. In this section we provide some heuristic strategies that allow us to cut down the computational effort by fixing some of the variables. It will also provide good-quality initial feasible solutions when solving, exactly, (MCSVM) using a commercial solver. Two different strategies are provided. The first one consists of applying a variable fixing strategy to reduce the number of h-variables in the model (originally n^2). The second approach consists of fixing to zero some of the z-variables. These nk variables allow us to model assignments between observations and classes. The proposed method is a math-heuristic approach, since after applying the adequate dimensionality reductions, Problem (MCSVM) (or (MCSVM$_{\text{RL}}$)) has to be solved. Also, although our strategies do not ensure any kind of optimality certificate, they produce a very good performance as will be shown in our computational experiments. Observe that when classifying datasets, the measure of the efficiency of a decision rule, as ours, is usually assessed by means of the accuracy of the classification on out-of-sample data, whereas the objective value of the proposed model is just an approximated measure of such an accuracy which cannot be computed only with the training data.

Algorithm 1 A math-heuristic approach.

1. Apply dimensionality reductions test based on algorithms 2 and 3.
2. Find an initial solution generating k separating hyperplanes.
3. Solve problem (MCSVM) (or (MCSVM$_{\text{RL}}$)) up to a prescribed accuracy for the train data.

In what follows we describe two strategies to reduce the dimensionality of the problem. These approaches are based on applying clustering techniques to the data. The methods are sensible to the number of clusters. For determining this parameter, we run a hierarchical clustering method, using as termination criterion a given squared Euclidean distance between the observations and their centroids.

2.4.1 Reducing the h-variables

Our first strategy comes from the fact that for a given observation x_i, there may be several possible choices for h_{ij} to assume the value one with the same final result. Recall that h_{ij} could be equal to one whenever x_j is a well-classified observation in the same class as x_i. The errors e_{ir} and d_{ir} are then computed by using the class of x_j but not the observation x_j itself. Thus, if a set of well-classified observations of the same class is close enough, only one of them can be the representative element of the group. In order to illustrate the procedure, we show in Figure 2.8 (left) a

4-classes and 24-points instance in which the classes are easily identified by applying any clustering strategy. In such a case (MCSVM) has $(24 \times 24 =)$ 576 h-variables, but if we allow h only to take value 1 at a single point in each cluster, we obtain the same result but reducing to 162 $(24 \times 6 + 18$, where the 18 comes from the observation mentioned in the formulation in which each well-classified observation can be a representative element of itself) the number of variables. In Figure 2.8 (right), we show some clusters based on the data, and a (random) selection of a unique point at each cluster for which the h-values are allowed to be one.

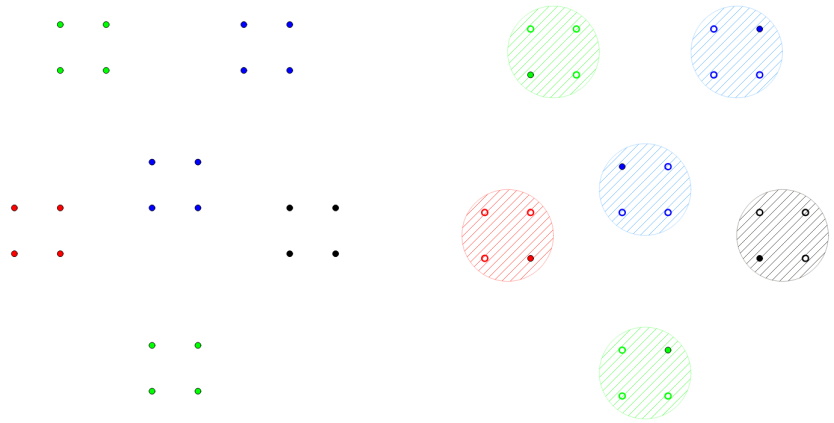


Figure 2.8: Clustering observations for reducing h-variables.

This strategy is summarized in Algorithm 2.

Algorithm 2 Strategy to reduce h-variables.

1. Cluster the dataset by *approximated* classes: $\mathcal{C}_1, \ldots, \mathcal{C}_c$.
2. Randomly choose a single point at each cluster, $x_{i_j} \in \mathcal{C}_j$, for $j = 1, \ldots, c$.
3. Set $h_{ij} = 0$ for $j \notin \{i_1, \ldots, i_c\}$.

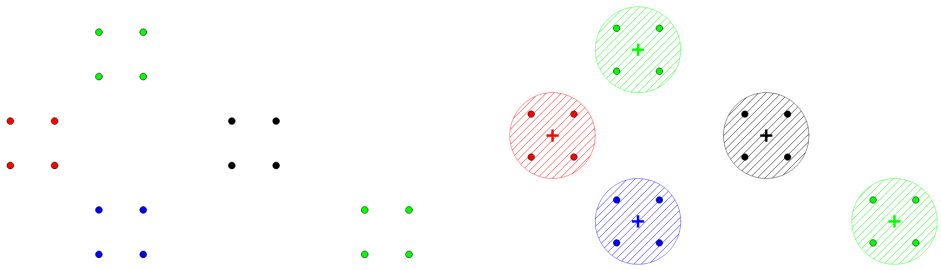


Figure 2.9: Illustration of the strategy to reduce the z-variables.

2.4.2 Reducing the z-variables

The second strategy consists of fixing to zero some of the point-to-class assignments (z-variables). In the picture shown in Figure 2.9 (left), one can see a set of points which seems reasonable to group in 5 clusters. One may notice that assignments from the red class to the black class (and vice versa) are rarely going to occur following our approach. This is due to the fact that given this configuration of points, our model would provide a cell for red points located far from a black cell (otherwise it would probably not be maximizing the distance between classes). Following this idea, we derived a procedure to fix some of the z-variables to zero. Another observation that comes from Figure 2.9, is that with respect to the red cluster we obtain the following sorting on the set of distances: $d^1_{green} \leq d_{blue} \leq d_{black} \leq d^2_{green}$. Then, since $d^1_{green} < d^2_{green}$, we may not take into account the distance to the green cluster on the very right. Thus, we would fix to zero all z_{is} variables that relate the red cluster with the maximum of their minimum distance set, that is, in this case we would fix to zero the z_{is}-variables associated to the black cluster with the red cluster ($d^1_{green} < d_{black}$ and $d_{blue} < d_{black}$) and vice versa.

The above observations lead us to some strategies for fixing z-variables to zero that we summarize in Algorithm 3.

Algorithm 3 Strategy to reduce z-variables.

1. Group the observations in L clusters, being each cluster formed by points of the same class. Let $a_{\ell s}$ be the centroids of the cluster, $\ell = 1, \ldots, L$, $s \in \{1, \ldots, k\}$.
2. Compute the squared Euclidean distance matrix between centroids: $\mathcal{D} = \left(\|a_{\ell s} - a_{q\bar{s}}\|^2\right)$.
3. For each cluster ℓ, $\ell = 1, \ldots, L$, assigned to class s, compute the cluster q with class $\bar{s} \neq s$ such that $\|a_{\ell s} - a_{q\bar{s}}\|^2$ is maximum and greater than a given threshold. For each observation i in cluster q, set $z_{is} = 0$. For each observation $\hat{\imath}$ in cluster ℓ, set $z_{\hat{\imath}\bar{s}} = 0$.

2.5 Experiments

In this section we report the results of our computational experience. We have run a series of experiments to analyze the performance of our model in both real and synthetic datasets. In what follows we describe the instances and the obtained results.

2.5.1 Real datasets

First, we study some real datasets widely used in the classification literature, and that are available in the UCI Machine Learning repository (Lichman et al. (2013)). Summarized information about the datasets is detailed in Table 2.1. In such a table we report, for each dataset, the number of observations (n), the number of features (p), the number of classes (k), the number of hyperplanes used in our separation (m), and the number of hyperplanes required by the OVO methodology (m_{OVO}).

Dataset	n	p	k	m	m_{OVO}
`Forest`	523	28	4	3	6
`Glass`	214	10	6	6	15
`Iris`	150	4	3	2	3
`Seeds`	210	7	3	2	3
`Waveform`	5000	21	3	2	3
`Wine`	178	13	3	2	3
`Zoo`	101	17	7	4	21

Table 2.1: Data sets used in our computational experiments.

For these datasets, we have run both the hinge-loss (MCSVM) and the ramp-loss (MCSVM_{RL}) models, measuring the margin with the ℓ_1 and the ℓ_2 norms. We have performed a 5-cross validation scheme to test each of the approaches. Thus, the data sets were split into 5 train-test random partitions. Then, the models were solved for the training sample and the resulting classification rule was applied to the test sample. We report the average accuracy, ACC, in percentage, of the 5 repetitions of the experiment on test:

$$\text{ACC} = \frac{\#\text{Well Classified Test Observations}}{\#\text{Test Observations}} \cdot 100.$$

The parameters of our models were chosen following a grid-based scheme. In particular, we calibrate the value of m (number of hyperplanes to be located) and the misclassification costs c_1 and c_2 in:

$$m \in \{2, \ldots, k\},\ c_1, c_2 \in \{0.0001, 0.001, 0.001, 0.1, 0.5, 1, 5, 10, 100, 1000, 10000\}.$$

The same methodology was also applied to the other methods: OVO, WW, CS and LLW, calibrating the misclassifying cost c in $\{10^i,\ i = -6, \ldots, 6\}$.

The Mathematical Programming models solving the MCSVM methods were coded in Python 3.6, and solved using Gurobi 7.5.2 on a PC Intel Core i7-7700 processor at 2.81 GHz and 16GB of RAM. The standard methods (OVO, WW and CS) were applied using R-KernLab. Finally, LLW was applied using the software package Lauer and Guermeur (2011).

In Table 2.2 we report the average accuracies obtained with our 4 models: ((MCSVM) and (MCSVM$_{\mathrm{RL}}$) with ℓ_1 and ℓ_2 norms) and those obtained with OVO, WW, CS and LLW. The first two columns (ℓ_1 RL and ℓ_1 HL) show the average accuracies of our two approaches (Ramp Loss - RL- and Hing Loss -HL-) using the ℓ_1-norm. On the other hand, the third and four columns (ℓ_2 RL and ℓ_2 HL) provide the same results for the ℓ_2-norm. In the last four columns, we report the average accuracies obtained with the OVO, WW, CS and LLW methods. The best accuracies obtained for each dataset are bolfaced in Table 2.2.

One can observe that our methods always outperform the results obtained by OVO, WW and CS, although the results are rather similar. Actually, running the two samples proportion test among them, we can not ensure significant differences in all cases. Comparing our methods with LLW the results are different. In four out of the 7 databases (`Forest`, `Glass`, `Iris` and `Wavefront`) our methods are superior to LLW with up to 10% significant differences with respect to the two samples proportion tests. In the remaining three databases (Seeds, Wine and Zoo) the results are similar with no statistical significant differences with respect to the two samples proportion test.

The results indicate that these UCI databases are friendly for linear classifiers (with the only exception of Glass) and thus all these methods perform reasonably well on test prediction. Thus, it is not possible to establish a clear ranking of these classification methods based only on these databases. In order to asses a more complete comparative of the methods we continue the analysis in the following subsection with a battery of more complex datasets.

Dataset	ℓ_1 RL	ℓ_1 HL	ℓ_2 RL	ℓ_2 HL	OVO	WW	CS	LLW
Forest	80.66	80.12	**82.30**	81.62	82.10	78.40	78.60	72.54
Glass	64.92	64.92	**65.32**	**65.32**	58.76	56.25	59.26	57.04
Iris	95.08	95.40	96.44	**96.66**	93.80	96.44	96.44	84.17
Seeds	93.66	93.66	93.52	93.52	91.02	93.52	93.52	**95.46**
Waveform	89.17	89.17	**91.27**	**91.27**	85.95	86.32	85.12	71.46
Wine	95.20	95.20	**96.82**	**96.82**	96.34	96.09	96.17	96.31
Zoo	89.75	89.75	89.75	89.75	87.44	87.68	87.68	**91.53**

Table 2.2: Average accuracies obtained for the real-world instances

2.5.2 Synthetic experiments

This section reports extra computational experiments over some synthetic instances that allow us to establish some rank of the methods based on their accuracies. We have generated 6 instances of 750 observations in $\mathbb{R}^{10}$ distributed as multivariate normal distributions separated by a constant factor. In addition to these, we have

also generated a bigger instance with 3000 observations belonging to 12 different classes. The instances are denoted as X`C`Y`N` where X is the number of classes (ranging in $\{2, 3, 4, 7, 10, 12\}$) and Y the number of different multivariate normal distributions (ranging in $\{4, 6, 8, 15, 20, 24\}$). All the instances are available at `http://bit.ly/SynthData_MCSVM` for benchmarking purposes. Observe that for each instance, the class labels have been randomly assigned to the normal distributions. For illustration purposes, a two-dimensional instance generated in the same way that our 10-dimensional instances is shown in Figure 2.10: the data are generated according to 20 normal distributions which are assigned to 10 classes.

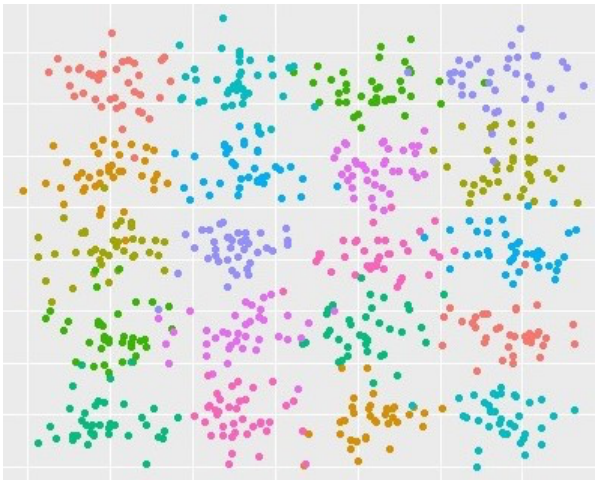


Figure 2.10: A 2-dimensional illustration of our instances.

In Table 2.3 we report the average accuracies obtained with a 10-fold cross validation experiment, in which 75 observations are taken into the training samples and 675 in the test samples for the first six instances, whereas 300 and 2700 observations are taken for the train and test samples in the last instance. As before, we have compared our approach (with the Euclidean norm and Hinge-Loss misclassification error) with the existing methodologies: OVO, WW, CS and LLW. The calibration of the parameters was also done as in the previous section.

Dataset	ℓ_2 HL	m	OVO	WW	CS	LLW
2C4N	**94.35**	2	60.75	60.75	60.75	60.75
3C6N	**85.74**	3	39.47	41.69	39.03	36.50
4C8N_1	**92.76**	4	36.46	32.37	29.14	31.86
4C8N_2	**91.78**	4	48.54	35.14	34.69	39.14
7C15N	**88.54**	6	27.37	19.64	18.63	20.35
10C20N	**85.81**	7	29.73	16.17	15.37	15.10
12C24N	**86.71**	8	28.02	18.14	13.13	14.19

Table 2.3: Average accuracies obtained for the synthetic instances.

One can observe in Table 2.3 that the results obtained with our approach are much better than those obtained with the other approaches. The generation procedure permits that, in the synthetic instances, separated clouds of points are assigned to the same class. As it can be anticipated, our methodology adapts well to this characteristic whereas the other approaches fail to handle these data. The reader may observe that this type of data are common in real-world datasets. In particular, many diseases are associated to low or high values of certain medical indices thus fitting to this topology in which separated clusters of observations belong to the same class.

Our main conclusion, from the results reported in Table 2.3, is that our method is adequate for this type of synthetic data, highly outperforming OVO, WW, CS and LLW. Moreover, the accuracy percentages are not only superior but they are also statistically better with respect to the two samples proportion test with a significance level of 1%.

2.6 Conclusions

In this chapter we propose a novel modeling framework for multiclass classification based on the Support Vector Machine paradigm, in which the different classes are linearly separated and the separation between classes is maximized. We propose two approaches, that depend on the way to account for the misclassification error, to compute an optimal arrangement of hyperplanes subdividing the space into cells, and so that each cell is assigned to a class based on the training data. The models result in Mixed Integer (Linear and Non Linear) Programming problems. Some dimensionality reduction and preprocessing strategies are presented in order to help solvers to find good (optimal) solutions of the corresponding problems. We also prove that an analogue of the kernel trick can be extended to this framework. The performance of this approach is illustrated on some well known datasets of the multi-category classification literature as well as in some synthetic, but still realistic, examples, in which our approach works remarkably well compared to the existing methodologies.

Chapter 3

Multisource hyperplanes location problem to fitting set of points

In this chapter we study the problem of locating a given number of hyperplanes minimizing an objective function of the closest distances from a set of points. We propose a general framework for the problem in which norm-based distances between points and hyperplanes are aggregated by means of ordered median functions. A compact Mixed Integer Linear (or Non Linear) programming formulation is presented for the problem and also an extended set partitioning formulation with an exponential number of variables is derived. We develop a column generation procedure embedded within a branch-and-price algorithm for solving the problem by adequately performing its preprocessing, pricing and branching. We also analyze geometrically the optimal solutions of the problem, deriving properties which are exploited to generate initial solutions for the proposed algorithms. Finally, the results of an extensive computational experience are reported. The issue of scalability is also addressed showing theoretical upper bounds on the errors assumed by replacing the original datasets by aggregated versions.

3.1 Introduction

Location Analysis deals with the determination of the *optimal* positions of facilities to satisfy the demand of a set of customers. The problems analyzed in the field are diverse but can be usually classified as: Discrete Location problems (DLP) and Continuous Location problems (CLP). In the first family, a set of potential facilities is previously given and the goal is to select, among them, the optimal ones under one or more criteria. The main tools for solving these problems come from Discrete Optimization, or more precisely, from Integer Linear Programming. In the second family of problems, the facilities have to be located in a continuous space and then, convex analysis and global optimization tools are needed to solve the problems. The most popular problem in the latter family is the Weber problem (Weber, 1909) in which a single point-facility has to be positioned on the plane so as to minimize the overall sum of the (Euclidean) distances to a set of (planar) demand points. The applications of both types of location problems are vast. DLP are more common in the location of *physical* facilities (as ATMs, supermarkets, stations, etc), while CLP are more useful when locating facilities in telecommunication networks (as wifi routers, servers, etc) or even to provide the sets of potential facilities for a DLP.

In this chapter we study a problem that falls into the family of CLP. More specifically, we focus on the determination of *optimal* hyperplanes fitting a given finite set of demand points. The location of a single hyperplane is a classical problem that has been addressed in different fields. On the one hand, this problem clearly extends the classical Weber problem, but where instead of locating zero-dimensional facilities one looks for locating higher dimensional structures. On the other hand, in

Statistics and Data Analysis, the determination of a hyperplane minimizing the sum of squares of vertical residuals is key for estimating a multivariate linear regression model using the Least Sum of Squares (Gauss (1877)). One can also find useful applications, both in Location Science and Data Analysis, for the problem of finding *optimal* hyperplanes fitting a set of points. For instance, Espejo and Rodríguez-Chía (2011) deals with the location of a rapid transit line on the plane to be used as an alternative transportation mean. Analogously, SVM is also based on constructing a hyperplane minimizing certain loss functions of the distances to a given set of points.

Scanning the literature one can find that most of the attention has been devoted to finding hyperplanes with any of the following assumptions (see e.g., (Martini and Schöbel, 1998; Schöbel, 2013, 2003, 2015; Martini and Schöbel, 2001; Brimberg et al., 2002, 2003; Blanco et al., 2018)): (a) the problem is embedded on the plane; (b) a single hyperplane has to be located; (c) the vertical distance between each point and the hyperplane is considered; or (d) the residuals are aggregated by the sum or the maximum operators. Our goal here is to study a generalization of this problem in which, we construct simultaneously a given number, m, of hyperplanes in any finite dimensional space, $\mathbb{R}^p$, by minimizing a rather general globalizing function, an ordered median function, of the residuals from the points to the fitting bodies. Ordered median functions aggregate the set of distances from the demand points to their closest hyperplanes (residuals) by means of a sorting, weighting averaging operation: distances are sorted and then their weighted sum is performed. The sum and maximum functions can be easily represented as ordered median functions with adequate choices of the weights inducing the median and center objective functions. Also, the k-centrum (sum of the k-th largest distances) or the cent-dian (convex combination of the sum and the max criteria) can be cast within this family of functions. In addition, different point-to-hyperplane norm-based distances are considered as a measure of the residuals of the fitting. Thus, this chapter naturally extends the analysis performed in Blanco et al. (2018) where the location of a single ordered median hyperplane was studied.

As in the classical Weber problem (Weber, 1909), the extension from the location of one to several facilities (the so-called multisource problem) is not trivial (Blanco et al., 2016). Actually, while the classical single-facility point location problem with standard distances (ℓ_p, polyhedral, etc) can be formulated as a Second Order Cone programming problem (Blanco et al., 2014) (being then polynomially solvable), its multisource version becomes a non-convex NP-hard problem (Blanco et al., 2016). In the case of locating hyperplanes, the situation is even harder, since the location of a single hyperplane is, in general, an NP-hard problem (see Blanco et al. (2018)) whose exact solution can only be obtained for vertical and polyhedral norm based residuals.

The problem considered in this chapter is not fully new although, in our opinion, it has not been fully analyzed and there is room for further improvement. In particular, similar problems have been analyzed from the Data Analysis field, and different names have been adopted. In the so-called *Clusterwise Linear Regression* (CLR) problem, a set of observations is provided and the goal is to cluster them by means of the sum of the squared residuals of several multivariate regression models (Späth, 1982; Hennig, 1999; Carbonneau et al., 2014; Park et al., 2017; Gitman et al., 2018). In (Bertsimas and Shioda, 2007), classification and regression are simultaneously performed, and also clustering by classical linear regression approaches. Finally, in (Bradley and Mangasarian, 2000), the clusters are constructed based on the closest distances to *optimal* hyperplanes in a given p-dimensional space. In the so-called *Piecewise Linear Regression* problem, a dependent variable is partitioned into j intervals and it adjusts linear bodies to each of them (see (McGee and Carleton, 1970)). However, only local search heuristic algorithms have been proposed for these problems, alternating clustering and regression techniques sequentially. Carbonneau et al. (2014) present a column generation algorithm for the (planar) clusterwise regression problem with sum of squared residuals which combined with some heuristic strategies outperforms previous results in the literature. Moreover, Park et al. (2017) generalized the clusterwise regression problem by allowing each entity to have more than one observation and propose an exact Mathematical Programming-based approach relying on column generation, and several heuristics.

In this chapter we provide a general framework for the simultaneous location of several hyperplanes to fit a data set using Mathematical Programming tools. We formulate the problem by using general norm-based error measures of the distance from points to hyperplanes and ordered median functions to aggregate the residuals. This approach generalizes both the standard multisource regression (Carbonneau et al., 2014; Park et al., 2017) and also the more recent proposal for the $m = 1$ case (Blanco et al., 2018).

The rest of the chapter is organized as follows. In section 3.2 we introduce the problem and fix the notation for the rest of the sections. This section also contains two illustrative examples taken from the literature. Section 3.3 is devoted to a first compact formulation for the problem. This formulation has a polynomial number of variables and constraints but its performance is not always good since it has a large integrality gap. For that reason, in section 3.4 we develop an alternative formulation with an exponential number of variables that is solved (exactly, for vertical and polyhedral-norm based residuals) within a branch-and-price (B&P) algorithm using column generation at each node of the branching tree. This section describes all the elements of this B&P: initialization, pricing (exact and heuristic) and branching. Section 3.5 reports our computational results based on two different datasets: the

classical 50 points dataset by Eilon et al. (1971) and another synthetic dataset randomly generated. Section 3.6 is devoted to explore scalability issues and finally section 3.7 draws some conclusions and future extensions.

3.2 Multisource location of hyperplanes

In this section we describe the problem under study and fix some set-related notation used in the chapter.

We are given a set of n observations/demand points (denoted as points from the rest of the chapter) in $\mathbb{R}^p$, $\{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ and $m \in \mathbb{Z}_+$ ($p > 0$). Our goal is to find m hyperplanes in $\mathbb{R}^p$ that minimize an objective function of the closest distances from points to hyperplanes. We denote the index sets of demand points and hyperplanes by $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$, respectively. Given $\omega \in \mathbb{R}^p$ and $\omega_0 \in \mathbb{R}$, we denote by $\mathcal{H}(\omega, \omega_0) = \{z \in \mathbb{R}^p : \omega' z + \omega_0 = 0\}$, the hyperplane in $\mathbb{R}^p$ with coefficients ω and intercept ω_0.

Several elements are involved when finding the *best* m hyperplanes to fit a set of demand points. In what follows we describe them:

- *Residuals:* The point-to-hyperplane measure of closeness. Given a demand point $x = (x_1, \ldots, x_p) \in \mathbb{R}^p$ and a hyperplane $\mathcal{H}(\omega, \omega_0)$, how far/close is the point from the hyperplane? The classical fitting methods use the so-called *vertical-distance* measure, which given a reference coordinate, say the p-th, computes the deviation $x_p + \frac{\omega_0}{\omega_p} + \sum_{\ell=1}^{p-1} \frac{\omega_\ell}{\omega_p} x_\ell$ for all $i \in I$, whenever $\omega_p \neq 0$. However, it has been already proposed that the use of more general distance measures based on norms may be advisable. In particular, some authors (see Blanco et al. (2020c)) have shown the usefulness of norm-based distances, such as polyhedral, or ℓ_p-distances ($p \geq 1$). Amongst them, we mention, for their importance, the Manhattan (ℓ_1-norm), the Tchebyshev (ℓ_∞-norm) or the Euclidean (ℓ_2-norm) distances.

  Thus, for a point $x \in \mathbb{R}^p$ and a hyperplane $\mathcal{H}(\omega, \omega_0)$, we consider the residual from x to $\mathcal{H}(\omega, \omega_0)$ as:

  $$\varepsilon_x(\omega, \omega_0) = \mathrm{D}\Big(x, \mathcal{H}(\omega, \omega_0)\Big) := \min\{\mathrm{D}(x, y) : y \in \mathcal{H}(\omega, \omega_0)\},$$

  where D is a norm-based distance or the vertical distance in $\mathbb{R}^p$ (see Mangasarian (1999); Blanco et al. (2018) for further details on this projection).

- *Allocation Rule*: Given a set of hyperplanes and a point, once the residuals to each of the hyperplanes are calculated, one has to allocate the point to a single hyperplane. Different alternatives can be considered, as the allocation to the

closest or the furthest hyperplane. In our framework we assume, as usual in Location Analysis, that each point is allocated to the hyperplane with the smallest residual, i.e., for a point $x \in \mathbb{R}^p$ and an arrangement of hyperplanes $\mathbb{H} = \Big\{\mathcal{H}(\omega_j, \omega_{0j}) : j \in J\Big\}$, the final residual point-to-hyperplanes is computed as:

$$\varepsilon_x\Big(\mathbb{H}\Big) = \min_{j\in J} \varepsilon_x(\omega_j, \omega_{0j}),$$

and the hyperplane, $\mathcal{H}(\omega, \omega_{0j})$, reaching such a minimum is the one where x is allocated (in case of ties among more than one hyperplane, a random assignment is performed).

- *Aggregation of Residuals*: Given a set of points and an arrangement of hyperplanes, once the residuals are computed with respect to the arrangement, and in order to find the m hyperplanes that best fit the n data points, a global measure of goodness must be chosen for aggregating the residuals. The classical aggregation functions are the sum or maximum of squared residuals. Most of these criteria can be cast within the framework of the family of ordered median aggregation criteria. More explicitly, given $x_1, \ldots, x_n \in \mathbb{R}^p$, an arrangement of hyperplanes $\mathbb{H} = \Big\{\mathcal{H}(\omega_j, \omega_{0j}) : j \in J\Big\}$, and $\lambda \in \mathbb{R}^n_+$ (with $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$), we can extend the definition presented in Chapter 1 and consider λ-ordered median function defined as:

$$\mathsf{OM}_\lambda(\varepsilon_1, \ldots, \varepsilon_n) = \sum_{i=1}^{n} \lambda_i \, e^\rho_{(i)}, \quad 1 \leq \rho < \infty \tag{OMF}$$

where $e_{(1)}, \ldots, e_{(n)}$ are defined such that $e_{(i)} \in \{\varepsilon_{x_1}(\mathbb{H}), \ldots, \varepsilon_{x_n}(\mathbb{H})\}$ for all $i \in I$ and $e_{(1)} \geq \cdots \geq e_{(n)}$. Observe that particular cases of ordered median problem for $\rho = 1$ (we consider $\rho = 1$ for the rest of the chapter) are the sum ($\lambda_i = 1$, $i = 1, \ldots, n$), the maximum ($\lambda_1 = 1$, $\lambda_i = 0$, $i \neq 1$), the k-centrum ($\lambda_i = 1$, $i = 1, \ldots, k$, $\lambda_j = 0$, $j > k$) or the ν-centdian, a convex combination of sum and max criterion ($\lambda_1 = 1, \lambda_i = \nu, i = 2, \ldots, n$), for $0 < \nu < 1$.

Summarizing all the above considerations, the Multisource Ordered Median Fitting Hyperplanes Problem (MOMFHP) can be stated as the problem of finding $\omega_1, \ldots, \omega_m \in \mathbb{R}^p$ and $\omega_{01}, \ldots, \omega_{0m} \in \mathbb{R}$ solving the following optimization problem:

$$\begin{aligned}
&\min \sum_{i=1}^{n} \lambda_i \, e_{(i)} && && \text{(MOMFHP}_0\text{)}\\
&\text{s.t. } e_i \geq \min_{j\in J} \varepsilon_{x_i}(\omega_j, \omega_{0j}), && \forall i = 1 \ldots, n,\\
&\quad \omega \in \mathbb{R}^p, \ \omega_{0j} \in \mathbb{R}, && \forall j = 1, \ldots, m,\\
&\quad e_i \geq 0, && \forall i = 1 \ldots, n.
\end{aligned}$$

where e_i represents the residual for the i-th point in the data set, for all $i = 1 \ldots, n$.

MOMFHP appears when different trends or clouds have to be differentiated on the demand points, and then, different hyperplanes want to be use to fitting to the points, such that the global error assumed, when the points are allocated to their closest hyperplanes, is as small as possible. In Figure 3.1 we illustrate a set of demand points in the plane which could be clustered into three groups according to different linear trends which are drawn in gray color.

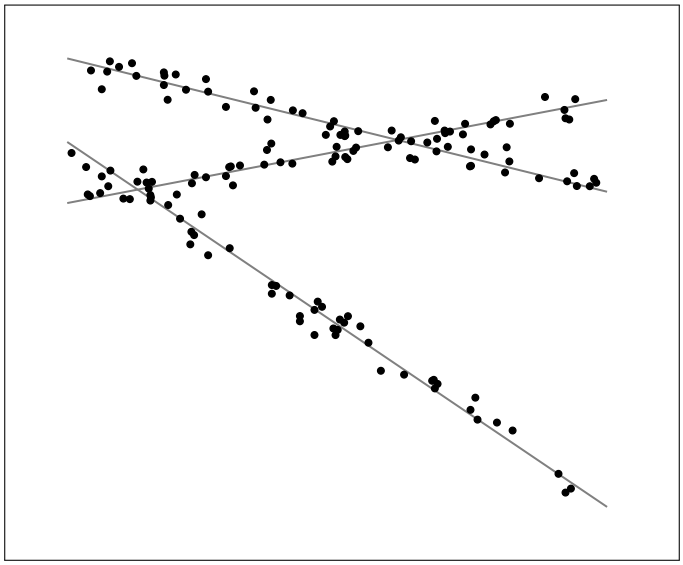


Figure 3.1: Illustration of a feasible solution of our problem for a set of demand points.

In the following example we illustrate the problem under analysis in two classical instances.

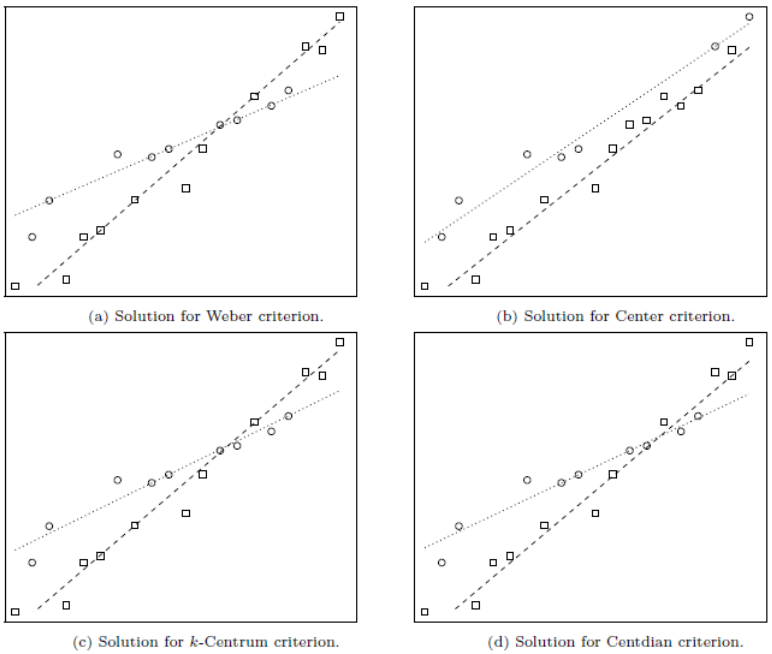
Example 3.1. *In the seminal paper by McGee and Carleton (1970), the authors illustrate the Clusterwise Linear Regression method with two instances. The first instance, (Quandt, 1958), consists of* 20 *points on the plane,* $\{x_1, \ldots, x_{20}\}$ *generated as follows:*

$$x_{i2} = 2.5 + 0.7x_{i1} + \epsilon_i, \ i = 1, \ldots, 12, \text{ and}$$
$$x_{i2} = 5 + 0.5x_{i1} + \epsilon_i, \ i = 13, \ldots, 20,$$

where ϵ *is randomly generated as a univariate normal distribution with mean* 0 *and standard deviation* 1*.*

We run our model with this dataset choosing as residuals the ℓ_1*-norm projection of the data onto the hyperplanes, and four different ordered median crite-*

ria: Weber, Center, $\lceil \frac{n}{2} \rceil$*-Centrum* $(\lambda = (\overbrace{1, \ldots, 1}^{\lceil \frac{n}{2} \rceil}, 0, \ldots, 0))$ *and* 0.9*-centdian* $(\lambda = (1, 0.9, \ldots, 0.9))$*. The results are shown in Figure 3.2.*



(a) Solution for Weber criterion.

(b) Solution for Center criterion.

(c) Solution for k-Centrum criterion.

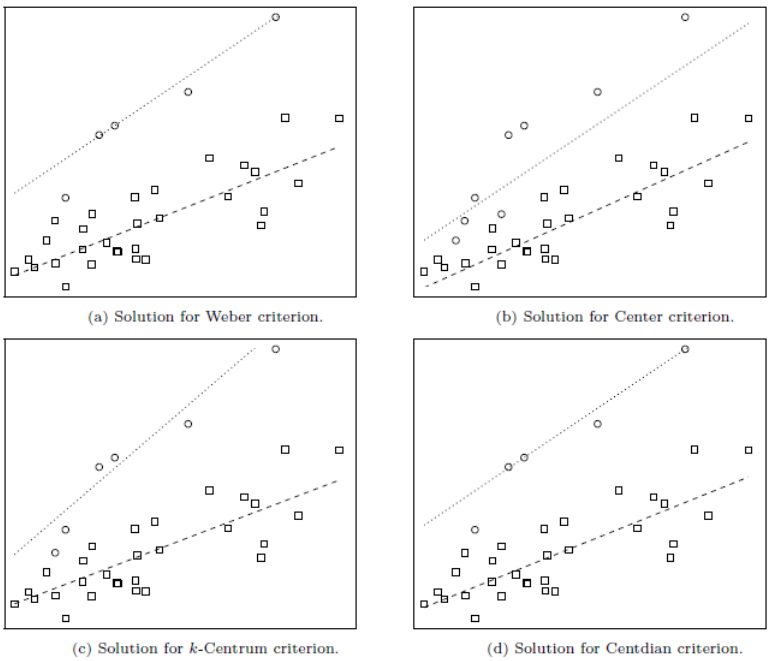
(d) Solution for Centdian criterion.

Figure 3.2: Lines obtained for Quandt dataset for ℓ_1-norm residuals and different criteria.

McGee and Carleton (1970) also analyzed a real instance, the `Boston` *dataset. It was motivated by the fact that regional stock exchanges were hurt by the abolition of give-ups in 1968. The model tries to analyze the dollar volume of sales on the Boston Stock Exchange with respect to dollar volumes for the New York and American Stock Exchanges, based on a dataset with* 35 *monthly observations from January 1967 to November 1969. One can observe, in the results shown in (Figure 3.3), that our models are able to adequately cast the trends of these observations.*

3.3 A compact formulation for (MOMFHP$_0$)

In this section we provide a Mathematical Programming formulation for (MOMFHP$_0$). The main components which involve decisions in this problem, and that have to be adequately included in a suitable formulation, are the representation of general norm-based residuals and the aggregation of residuals using an ordered median function. We describe here how to incorporate all these elements into a Mathematical Programming formulation which in many cases is suitable to be solved with any of the available MIP/MINLP solvers.



(a) Solution for Weber criterion.

(b) Solution for Center criterion.

(c) Solution for k-Centrum criterion.

(d) Solution for Centdian criterion.

Figure 3.3: Lines obtained for Boston dataset for ℓ_1-norm residuals and different criteria.

Theorem 3.1. *Let* $\{x_1, \ldots, x_n\} \subseteq \mathbb{R}^p$, $m \in \mathbb{Z}_+$ $(m > 0)$ *and* $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$. *Then,* (MOMFHP$_0$) *can be equivalently reformulated as follows:*

$$\min \sum_{k=1}^{n} u_k + \sum_{i=1}^{n} v_i \qquad \text{(MOMFHP)}$$

$$\begin{aligned}
s.t.\ & u_k + v_i \geq \lambda_k e_i, && \forall i, k = 1, \ldots, n, && (3.1)\\
& e_i \geq \varepsilon_{x_i}(\omega_j, \omega_{0j}) - M_{ij}(1 - z_{ij}), && \forall i = 1, \ldots, n,\ j = 1, \ldots, m, && (3.2)\\
& \sum_{j=1}^{m} z_{ij} = 1, && \forall i = 1, \ldots, n, && (3.3)\\
& z_{ij} \in \{0, 1\}, && \forall i = 1, \ldots, n,\ j = 1, \ldots, m, && (3.4)\\
& e_i \in \mathbb{R}_+, && \forall i = 1, \ldots, n, && (3.5)\\
& \omega_j \in \mathbb{R}^p, \omega_{0j} \in \mathbb{R}, && \forall j = 1, \ldots, m, && (3.6)\\
& u_k, v_i \in \mathbb{R}, && \forall i, k = 1, \ldots, n. && (3.7)
\end{aligned}$$

where M_{ij} *are upper bounds on the residual values* $\varepsilon_{x_i}(\omega_j, \omega_{0j})$, $\forall i = 1, \ldots, n, j = 1, \ldots, m$.

Proof. First, observe that given a set of residuals $e_1, \ldots, e_n \geq 0$, the evaluation of the objective function in (MOMFHP$_0$) requires sorting and averaging them (the

residuals) with the λ-weights. In Blanco et al. (2014), the authors proved that the computation of $\sum_{k=1}^{n} \lambda_k e_{(k)}$ can be done by means of the optimal value of the following Linear Programming Problem (see Blanco et al. (2014)):

$$\sum_{k=1}^{n} \lambda_k e_{(k)} = \left\{ \begin{array}{ll} \min & \sum_{k=1}^{n} u_k + \sum_{i=1}^{n} v_i \\ \text{s.t.} & u_k + v_i \geq \lambda_k e_i, \quad \forall k, i = 1, \ldots, n, \\ & u_k, v_i \in \mathbb{R}, \quad \forall k, i = 1, \ldots, n. \end{array} \right.$$

Thus, the objective function in (MOMFHP$_0$) can be replaced by the above objective function and the constraints incorporated to the rest of constraints in the model.

In order to identify the point-to-hyperplane allocation we consider the following set of binary variables:

$$z_{ij} = \left\{ \begin{array}{ll} 1 & \text{if the } i\text{-th observation is assigned to } \mathcal{H}(\omega_j, \omega_{0j}), \\ 0 & \text{otherwise,} \end{array} \right.$$

$\forall i = 1, \ldots, n, j = 1, \ldots, m$.

Note that with our allocation rule, an observation can be always assigned to a hyperplane that reaches the minimum residual among all the possible assignments to the m hyperplanes.

Finally, using the variables previously described, the objective function computes the ordered median function of the residuals. Constraints (3.2) assure the correct definition of the residuals e_i and the allocation to their correct hyperplane. Indeed, if $z_{ij} = 1$ this constraint forces e_i to take the value of $\varepsilon_{x_i}(\omega_j, \omega_{0j})$. Constraints (3.3) assure that only one of these variables will be equal to 1, which in turns forces by the minimization character of the objective function to be the one with the correct assignment. Finally, (3.4)–(3.7) are the domains of the variables. □

Remark 3.1. *Observe that the different choices of ordered median functions are embedded into constraint* (3.1)*. In some particular cases, this formulation can be simplified avoiding useless variables and constraints.*

- **m-Median Problem** *($\lambda = (1, \ldots, 1)$) In this case, since the ordering does not affect the aggregation operator, the u and v-variables can be avoided, and the problem simplifies to:*

$$\begin{array}{ll} \min & \sum_{i=1}^{n} e_i \\ \text{s.t.} & (3.2) - (3.6). \end{array}$$

- m-**Center Problem** *($\lambda = (1, 0, \ldots, 0)$): For the Center problem, one can represent the objective function, $\max_{i \in I} e_i$, by using an auxiliary variable, t, in the usual manner:*

$$\begin{aligned} \min\ & t \\ \text{s.t. } & (3.2) - (3.6), \\ & t \geq e_i, \qquad \forall i = 1, \ldots, n. \end{aligned}$$

- m-k-**Center Problem** *($\lambda = (\overbrace{1, \ldots, 1}^{k}, 0, \ldots, 0)$): For the m-Centrum problem, in Ogryczak and Tamir (2003) the authors derive a formulation similar to the one for the center problem:*

$$\begin{aligned} \min\ & k\,t + \sum_{i=1}^{n} r_i \\ \text{s.t. } & (3.2) - (3.6), \\ & r_i \geq e_i - t, \qquad \forall i = 1, \ldots, n, \\ & t \geq 0, \\ & r_i \geq 0, \qquad \forall i = 1, \ldots, n. \end{aligned}$$

Note also that the explicit expression of $\varepsilon_{x_i}(\omega_j, \omega_{0j})$ and then, the difficulty of the optimization problem above, depends (apart from the binary variables that appears in the problem) on the choice of the distance measure D which defines the residuals of the fitting. In what follows we describe general shapes for the distances inducing the residuals and how they can be incorporated to (MOMFHP).

3.3.1 Vertical Distance Residuals

Although not rigorously a distance measure, the so-called *vertical distance* is a very common measure for computing the residuals in Data Analysis. The vertical distance is computed as the absolute deviation, with respect to one of the coordinates, of the hyperplane. Without loss of generality, we consider that the deviation is computed with respect to the p-th coordinate, and then, one can assume that $\omega_{jp} = -1$ for $j \in J$. Given $x \in \mathbb{R}^p$ and an hyperplane $\mathcal{H}(\omega, \omega_0)$ the vertical distance residual is calculated as:

$$\varepsilon_x(\omega, \omega_0) = \left| x_p - \omega_0 - \sum_{\ell=1}^{p-1} \omega_\ell x_\ell \right|.$$

This measure can be incorporated to (MOMFHP), replacing (3.2) by the follow-

ing set of linear constraints:

$$e_i \geq x_{ip} - \omega_{0j} - \sum_{\ell=1}^{p-1} \omega_{j\ell} x_{i\ell} - M_{ij}(1 - z_{ij}), \forall i = 1, \ldots, n,\ j = 1, \ldots, m,$$
$$e_i \geq -x_{ip} + \omega_{0j} + \sum_{\ell=1}^{p-1} \omega_{j\ell} x_{i\ell} - M_{ij}(1 - z_{ij}), \forall i = 1, \ldots, n,\ j = 1, \ldots, m, .$$

Thus, becoming (MOMFHP) a Mixed Integer Linear Programming problem.

Remark 3.2 (Support Vector Regression)**.** *One particular case in which vertical residuals are used in Machine Learning tools is in Support Vector Regression. Drucker et al. (1997) proposed this methodology for obtaining regression models based on Support Vector Machines. The method is based on fitting a hyperplane to the set of points* $\{x_1, \ldots, x_n\}$ *with a modified vertical distance, such that only the residuals greater than a given threshold* $\Delta \geq 0$ *are accounted for, apart from maximizing the separation between the observations at each of the sides of the hyperplanes. SVR can be modeled as follows:*

$$\begin{aligned}
\min\ & \frac{1}{2}\|\omega\|_2^2 + c\sum_{i=1}^{n} e_i & \\
\text{s.t. } & e_i \geq \left| x_{ip} - \sum_{\ell=1}^{p-1} \omega_\ell x_{i\ell} - \omega_0 \right| - \Delta, & \forall i = 1, \ldots, n, \\
& \omega \in \mathbb{R}^{p-1}, \omega_0 \in \mathbb{R}, & \\
& e_i \geq 0, & \forall i = 1, \ldots, n,
\end{aligned}$$

where c *is a given parameter.*

Observe that the measure used in this approach is nothing but a truncated version of the vertical distance:

$$\varepsilon_x(\omega, \omega_0) = \begin{cases} |x_p - \alpha - \sum_{\ell=1}^{p-1} \omega_\ell x_\ell| & \text{if } |x_p - \omega_0 - \sum_{\ell=1}^{p-1} \omega_\ell x_\ell| > \Delta, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, this shape of the residuals can also be embedded in our multisource framework, just by adding to the objective functions the terms measuring the norms of the coefficients of the hyperplanes, i.e., replacing the objective function in (MOMFHP) *by*

$$\frac{1}{2}\sum_{j=1}^{m} \|\omega_j\|_2^2 + \sum_{i=1}^{n} \lambda_i\, e_{(i)}.$$

In case $m = 1$ *and* $\lambda = (1, \ldots, 1)$*, we obtain classical SVR taking also into account*

the parameter Δ, but more flexible counterparts can be generated with our framework.

3.3.2 Norm-based Residuals

For general norm-based distances, a given observation $x \in \mathbb{R}^p$ and a set of m hyperplanes defined by $\omega_1, \ldots, \omega_m \in \mathbb{R}^p$ and $\omega_{01}, \ldots, \omega_{0m} \in \mathbb{R}$ inducing the arragement $\mathbb{H} = \left\{\mathcal{H}(\omega_j, \omega_{0_j}) : j \in J\right\}$, based on (Mangasarian, 1999, Theorem 2.1), the projection, $\hat{x}$, of x consistent with the residual ε induced by a norm $\|\cdot\|$ is

$$\hat{x} = x_{-0} - \min_{j \in J} \frac{\omega_{0j} + \omega_j' x}{\|(\omega_{j1}, \ldots, \omega_{jp})\|_*} \kappa(\omega_j),$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$ and $\kappa(\omega) = \arg \max_{\|z\|=1} (\omega_{j1}, \ldots, \omega_{jp})' z$. Moreover, the residuals can be written as:

$$\varepsilon_x(\mathbb{H}) = \min_{j \in J} \frac{|\omega_{0j} + \omega_j' x|}{\|(\omega_{j1}, \ldots, \omega_{jp})\|_*}. \tag{3.8}$$

Remark 3.3 (ℓ_1-norm case)**.** *In the case of the ℓ_1-norm residuals, the expression above reduces to:*

$$\varepsilon_{x_i}(\omega_j, \omega_{0j}) = \min_{j \in J} \frac{|\omega_{0j} + \omega_j' x_i|}{\max_{\ell=1,\ldots,p} |\omega_{j\ell}|}, \tag{3.9}$$

and constraints (3.2) *can be replaced in* (MOMFHP) *by:*

$$e_i \geq \omega_{0j} + \sum_{\ell=1}^{p} \omega_{j\ell} x_{i\ell} - M_{ij}(1 - z_{ij}), \qquad \forall i = 1, \ldots, n,\ j = 1, \ldots, m, \tag{3.10}$$

$$e_i \geq -\omega_{0j} - \sum_{\ell=1}^{p} \omega_{j\ell} x_{i\ell} - M_{ij}(1 - z_{ij}), \qquad \forall i = 1, \ldots, n,\ j = 1, \ldots, m, \tag{3.11}$$

$$\omega_{j\ell} = \eta_{j\ell}^+ - \eta_{j\ell}^-, \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.12}$$

$$\eta_{j\ell}^+ \leq U_{j\ell}\, \xi_{j\ell}, \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.13}$$

$$\eta_{j\ell}^- \leq U_{j\ell}\, (1 - \xi_{j\ell}), \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.14}$$

$$\theta_{j\ell} = \eta_{j\ell}^+ + \eta_{j\ell}^-, \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.15}$$

$$\theta_{j\ell} \leq 1, \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.16}$$

$$\theta_{j\ell} \geq \mu_{j\ell}, \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.17}$$

$$\sum_{\ell=1}^{p} \mu_{j\ell} = 1, \qquad \forall j = 1, \ldots, m, \tag{3.18}$$

$$\eta_{j\ell}^+, \eta_{j\ell}^-, \theta_{j\ell} \in \mathbb{R}_+^p, \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.19}$$

$$\mu_{j\ell}, \xi_{j\ell} \in \{0, 1\}, \qquad \forall j = 1, \ldots, m,\ \ell = 1, \ldots, p, \tag{3.20}$$

where M_{ij} and $U_{j\ell}$ are big enough constants.

We have introduced in the above formulation some new variables to model the ℓ_∞-distance in the denominator of the residual (3.9)*. In particular, for each $j = 1, \ldots, m$, the p-dimensional variable $\boldsymbol{\theta}_j$ models the vector $(|\omega_{j1}|, \ldots, |\omega_{jp}|)$ for which the maximum has to be taken; $\eta^+_{j\ell}$ represents $\max\{\omega_{j\ell}, 0\}$ and on the other hand $\eta^-_{j\ell}$ the amount $\max\{-\omega_{j\ell}, 0\}$, for all $\ell = 1, \ldots, p$. Clearly, one has that$\omega_j = \boldsymbol{\eta}^+_j - \boldsymbol{\eta}^-_j$ and $\boldsymbol{\theta}_j = \boldsymbol{\eta}^+_j + \boldsymbol{\eta}^-_j$ as imposed in constraints* (3.12)-(3.15)*, where the auxiliary variables ξ enforce that for each coordinate, either the positive or the negative part assumes value zero (avoiding other types of decompositions). Constraints* (3.16)*,* (3.17) *and* (3.18) *assure that $\max_{j\in J} |\omega_j| = 1$ via the auxiliary binary variables $\mu_{j\ell} \in \{0, 1\}$ that take value 1 in exactly one position (the one where the maximum is achieved).*

Thus, the formulation assures that $\max_{l=1,\ldots,p} |\omega_{j\ell}| = 1$, and then, the expression of the residual becomes $\varepsilon_{x_i}(\omega_j, \omega_{0j}) = \min_{j\in J} |\omega_{0j} + \omega'_j x_i|$ for all $i = 1, \ldots, n$, and $j = 1, \ldots, m$.

In this case, also (MOMFHP) *becomes a Mixed Integer Linear Programming problem.*

3.4 Set partitioning formulation

In this section we alternatively reformulate (MOMFHP$_0$) as a set partitioning problem (SPP) (see e.g., Balas and Padberg (1976)). Our SPP is based on the idea that once the m clusters of demand points are known , (MOMFHP$_0$) reduces to finding the optimal hyperplanes for each of those clusters in which all the residuals are aggregated by means of an ordered median function. In particular, let S be a cluster of observations $S \subseteq I$. We denote by e_S the cost of cluster S, i.e. the overall aggregation of the residuals of the data in S, and for each $i \in S$, e^i_S the marginal contribution of observation i in the cluster ($e_S = \sum_{i\in S} e^i_S$). Finally, we define the variables

$$y_S = \begin{cases} 1 & \text{if cluster } S \text{ is selected,} \\ 0 & \text{otherwise} \end{cases}, \quad \forall S \subseteq I.$$

The set partitioning formulation for (MOMFHP$_0$) is:

$$\min \sum_{i\in I} \sum_{S\subseteq I: S\ni i} \lambda_i e_S^{(i)} y_S \tag{3.21}$$

$$\text{s.t.} \sum_{S\subseteq I} y_S = m, \tag{3.22}$$

$$\sum_{\substack{S\subseteq I:\\ S\ni i}} y_S = 1, \qquad \forall\, i \in I, \tag{3.23}$$

$$y_S \in \{0,1\}, \qquad \forall S \subseteq I. \tag{3.24}$$

where $e_S^{(i)}$ is the i-th element in the sorted sequence of (active) residuals. In the above formulation the objective function computes the ordered median aggregation of the residuals (each demand point i allocated to its cluster S). Constraint (3.22) assures that m clusters have to be computed and constraints (3.23) that each observation belongs to a single cluster.

In the same manner that we formulate the ordered median objective function in the compact formulation we can equivalently reformulate the problem above as follows:

$$\min \sum_{k=1}^{n} u_k + \sum_{i=1}^{n} v_i \tag{3.25}$$

$$\text{s.t.} \quad u_k + v_i \geq \lambda_k \sum_{S\ni k} e_S^i y_S, \qquad \forall i,k \in I, \tag{3.26}$$

$$\sum_{S\subseteq I} y_S = m, \tag{3.27}$$

$$\sum_{\substack{S\subseteq I:\\ S\ni i}} y_S = 1, \qquad \forall\, i \in I, \tag{3.28}$$

$$y_S \in \{0,1\}, \qquad \forall S \subset I,$$

$$u_k, v_i \in \mathbb{R}, \qquad \forall i,k \in I.$$

This problem will be referred to as the *Master Problem*.

The problem above, although adequately solves the problem of finding the m hyperplanes once the optimal clusters are computed, has an exponential number of variables (and coefficients to incorporate to constraints (3.26)), and then it is hard to solve unless the number of points is very small. Thus, we propose a column generation (CG) approach for solving, efficiently, the problem above by adding new variables to the model as needed and not considering all of them at the same time. A pseudocode indicating the general procedure is shown in Algorithm 4.

Initially, a (small) subset of the y-variables is considered (those indexed by the sets in $\mathcal{S}_0$) and a relaxed version of the problem is solved with only these variables.

It implies to compute the amounts e_S^i for all $S \in \mathcal{S}_0$ and $i \in S$. Next, it has to be checked whether the optimality condition is satisfied. If it is not the case, a new set of variables is found and added to the relaxed problem and the procedure is repeated.

Algorithm 4 General Scheme for the CG approach.

Data: $\{x_1, \ldots, x_n\} \subseteq \mathbb{R}^p$, $m \in \mathbb{Z}_+$ $(m > 0)$, $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$.

1. **Preprocessing:** Compute a set of initial solutions for the problem $\mathcal{S}_0 = \{S_1, \ldots, S_K\}$ with $S_k \subseteq I$ for all $k = 1, \ldots, K$.

2. **Relaxed Master:** Solve the relaxed master problem:

$$\begin{aligned} \min & \sum_{k=1}^{n} u_k + \sum_{i=1}^{n} v_i & & \\ \text{s.t. } & u_k + v_i \geq \lambda_k \sum_{S \ni k} e_S^i y_S, & & \forall i, k \in I, \\ & \sum_{S \in \mathcal{S}_0} y_S = m, & & \\ & \sum_{\substack{S \in \mathcal{S}_0: \\ S \ni i}} y_S = 1, & & \forall\, i \in I, \\ & 0 \leq y_S \leq 1, & & \forall S \in \mathcal{S}_0. \end{aligned} \tag{RMP}$$

3. **New Columns** : Check if new columns have to be added to (RMP).

   If: Optimality is satisfied $\mathcal{C}^* = \{S \subseteq I : y_S^* = 1\}$.
   Else: Update $\mathcal{S}_0$ with the new columns and go to **2.**
   End

Result:$\{\mathcal{H}(\omega_S, \omega_{0S}) : S \in \mathcal{C}^*\}$.

The crucial steps in the implementation of the CG approach are the following:

1. *Preprocessing:* Generation of initial feasible solutions in the form of initial clusters (and their costs). This step may be improved by the theoretical properties verified by the corresponding optimal hyperplanes. We have implemented different initial solutions based on properties of the optimal solution of median and center hyperplanes (see Section 3.4.2).

2. *Pricing:* As already mentioned, in set partitioning problems, instead of solving initially the problem with the whole set of exponentially many variables, the variables have to be incorporated *on-the-fly* by solving adequate pricing

subproblems derived from previously computed solutions until the optimality of the solution is guaranteed.

3. *Branching:* The rule that creates new nodes of the branch and bound tree when a fractional solution is found at a node of the search tree. In this problem, we have adapted the Ryan-and-Foster branching complemented by a secondary ad-hoc branching in some special situations.

3.4.1 Preprocessing

In the preprocesing phase, we generate different types of initial solutions, which implies the initialization of the CG algorithm with a given set of variables.

We consider different types of initial solutions derived from the construction of hyperplanes fitting the sets of points. First, to initialize the pool of columns, $\mathcal{S}_0$, we randomly generate hyperplanes passing through p original points. Among the various strategies compared, we have eventually implemented one that performs completions with $p-2$ points of all possible couples of original points. This strategy augment $\frac{n(n-1)}{2}$ new variables into the pool. In addition, we also augment to the pool the best single hyperplane that fits all the points, assuring that the problem is feasible at the root node of the branch-and-price tree. Finally, apart from the above initial columns, we also charge an initial heuristic solution (in the y-variables) so as to have a good upper bound in our branch-and-price algorithm. Our algorithm chooses at random m mutually disjoint subsets of p points and finds the hyperplanes determined by those m sets of p points. Next, we perform a 1-interchange heuristic generating a new hyperplane that replaces, one at a time, one of those currently considered in the configuration until the first iteration where no improvement is possible. The incumbent set of hyperplanes and their corresponding allocations is considered an initial solution that is loaded into the solver.

3.4.2 Median and center optimal hyperplanes

We have used the following properties to build the initial solutions of our CG approach since they determine optimal hyperplanes for specific objective functions, see e.g., Schöbel (2003).

Lemma 3.1. *The following properties are verified:*

1. Weak incidence property*: There exists an optimal median hyperplane passing through p affinely independent points.*

2. Pseudo-halving property*: Every optimal median hyperplane, $\mathcal{H}(\omega^*, \omega_l^*)$ verifies*

$$\#\Big\{i \in I : x_i \in \mathcal{H}^-(\omega^*, \omega_0^*)\Big\} \leq \tfrac{n}{2} \text{ and } \#\Big\{i \in I : x_i \in \mathcal{H}^+(\omega^*, \omega_0^*)\Big\} \leq \tfrac{n}{2}.$$

3. Weak blockedness property*: There exists an optimal center hyperplane that is at maximum distance from $p+1$ of the points.*

4. Parallel facets property*: There exists an optimal center hyperplane that is parallel to a facet of the convex hull of the given points.*

For the more general ordered median objective function, we have proved the following result that characterizes the ordered median hyperplanes. In what follows, we derive a novel result for these hyperplanes that will be useful in the preprocessing phase of our CG approach.

Let us introduce the following notation:

- Let $\mathcal{B}$ be the subdivision of $\mathbb{R}^{p+1}$ induced by the following arrangement of hyperplanes:

$$B_{ij}^{ab} = \Big\{(\omega, \omega_0) \in \mathbb{R}^{p+1} : a(\omega' x_i + \omega_0) = b(\omega' x_j + \omega_0)\Big\}, \forall i, j \in I, a, b \in \{-1, 1\}.$$

- Let $\mathcal{S}$ be the subdivision of of $\mathbb{R}^{p+1}$ induced by the following arrangement of hyperplanes

$$S_i = \Big\{(\omega, \omega_0) \in \mathbb{R}^{p+1} : \omega' x_i + \omega_0 = 0\Big\}, \forall i \in I.$$

Lemma 3.2. *If $\mathcal{H}(\omega, \omega_0)$ is an optimal ordered median hyperplane then (ω, ω_0) is an extreme point of a cell in the subdivision of $\mathbb{R}^{p+1}$ induced by the intersection $\mathcal{B} \cap \mathcal{S}$.*

Proof. For a given hyperplane $\mathcal{H}(\omega, \omega_0)$, let us consider the objective function of the problem, namely $\sum_{i \in I} \lambda_i e_{(i)}$, where $e_i = D(\mathcal{H}(\omega, \omega_0), x_i)$.

It is clear that within each cell of the subdivision $\mathcal{B}$ the sorting of the residuals does not change. In addition, in each cell of the subdivision $\mathcal{S}$ the sign of $\omega' x_i + \omega_0$ is either positive or negative (but does not change) for each $i \in I$. Therefore, if $C \in \mathcal{B} \cap \mathcal{S}$ is a cell in the subdivision induced by $\mathcal{B} \cap \mathcal{S}$, there is a permutation σ that fixes the sorting of the residuals and also a constant vector $(sign(\omega' x_1 + \omega_0), \ldots, sign(\omega' x_n + \omega_0)) \in \{-1, 1\}^n$ such that

$$\sum_{i \in I} \lambda_i e_{(i)} = \sum_{i \in I} \lambda_i \frac{sign(\omega' x_i + \omega_0)(\omega' x_i + \omega_0)}{\|\omega\|_*} = \frac{\sum_{i \in I} \lambda_i sign(\omega' x_i + \omega_0)(\omega' x_i + \omega_0)}{\|\omega\|_*}.$$

The above function is the ratio of a non-negative linear function and a convex function, then it is quasiconcave provided that $(\omega, \omega_0) \in C$. Therefore, it attains its minima at the extreme points of this region. Hence, if $\mathcal{H}(\omega, \omega_0)$ is an optimal ordered median hyperplane (ω, ω_0) must be an extreme point of some of those cells. □

The above result allows us to interpret optimal ordered median hyperplanes also in terms of a geometrical description as those that meet p conditions between the following cases: i) passing through points $x_i,\ i \in I$, and ii) being at the same distance of two points $x_i, x_j,\ i, j \in I$. Optimal ordered median hyperplanes must also satisfy, for some $k = 1, \ldots, p$, the following property: it contains k points $x_i,\ i \in I$ and it is at the same distance from $p - k$ pairs $x_i, x_j\ i, j \in I$.

In our computational results we have computed the initial solutions and the initial pool of variables for the objective functions of type median, k-centrum and centdian, using the weak incidence property, whereas for the center objective function we use the weak blockedness property.

3.4.3 Pricing problem

Certifying optimality in a CG approach avoiding the inclusion of all the columns into the relaxed master problem, (RMP), requires testing whether a new tentative column must be added to the problem. In case no new candidates are added to the master problem, the optimality is guaranteed, otherwise, one should add the new columns and repeat the process (Step 3 in Algorithm 4). Searching for new columns to be added to the model will be performed by looking at the dual formulation of the set partitioning formulation.

Let γ be the dual variable for constraint (3.27), ϕ_i the dual variables for constraints (3.28) and δ_{ik} the dual variables for constraints (3.26). Then, the dual of the Master Problem is the following:

$$
\begin{aligned}
&\max -m\gamma + \sum_{i=1}^{n} \phi_i \\
&\text{s.t.} \ \sum_{k=1}^{n} \delta_{ik} = 1, && \forall i \in I, \\
&\quad \sum_{i=1}^{n} \delta_{ik} = 1, && \forall k \in I, \\
&\quad -\sum_{i \in S} \sum_{k=1}^{n} \lambda_k e_S^i \delta_{ik} - \gamma + \sum_{i \in S} \phi_i \leq 0, && \forall S \subseteq \mathcal{S}_0, \\
&\quad \delta_{ik}, \gamma, \phi_i \geq 0.
\end{aligned}
$$

Hence, for any $S \subset \mathcal{S}_0$, since y_S does not appear in the objective function, the reduced cost for variable y_S is:

$$\bar{e}_S = \gamma - \sum_{i \in S} \phi_i + \sum_{i \in S} \sum_{k=1}^{n} \lambda_k \delta_{ik} e_S^i.$$

Then, given an optimal dual solution $(\gamma^*, \phi^*, \delta^*)$, and considering the binary variables

$$w_i = \begin{cases} 1 & \text{if the } i\text{th observation is chosen for the set } S \text{ indexing the new column,} \\ 0 & \text{otherwise,} \end{cases}$$

the pricing problem is to choose the subset S with minimum reduced cost, i.e., to solve:

$$\begin{aligned} \min & -\sum_{i=1}^{n} \phi_i^* w_i + \gamma^* + \sum_{i=1}^{n} c_i^* r_i \\ \text{s.t. } & z_i \geq \varepsilon_{x_i}(\omega, \omega_0)), && \forall i \in I, \\ & r_i \geq z_i - M(1 - w_i), && \forall i \in I, \\ & w_i \in \{0, 1\}, && \forall i \in I, \\ & z_i, r_i \geq 0, && \forall i \in I, \\ & \omega \in \mathbb{R}^p, \omega_0 \in \mathbb{R}, \end{aligned}$$

where $c_i^* = \sum_{k=1}^{n} \lambda_k \delta_{ik}^*$, $\forall = 1, \ldots, n$.

If the optimal value of this problem is negative, the new column $y_{\hat{S}}$ is added to the pool, where $\hat{S} = \{i : w_i = 1\}$, since its reduced cost in the (RMP) is negative, and thus, it improves the objective function of the master problem. Otherwise, optimality is certified and we are finished.

Heuristic pricing

The exact pricing routine described above is an NP-hard problem and thus in general, it takes time finding new columns to be added to the pool or to certify optimality of the reduced master problem. This last task cannot be avoided, provided that we design an exact solution algorithm. Nevertheless, in many occasions finding promising new variables can be done at very low computational time resorting to heuristic schemes.

In our problem, we propose to test hyperplanes chosen from a discrete set of potential candidates. To do so, we set a p-dimensional grid on the normalized space of ω_0 and ω coefficients. Each point represents a hyperplane to be tested. Once the candidate (ω, ω_0) is chosen we determine which set of points S is going to be added to the new variable y_S. This is done with a simple greedy rule: choose those points with negative reduced cost with respect to $H(\omega, \omega_0)$.

If after this process we find a hyperplane that produces a negative reduced cost, we add this new column to the pool. Otherwise, we proceed with the exact pricer. This scheme speeds up the search for new columns without loosing the exactness of

the whole algorithm.

3.4.4 Branching

The set partitioning formulation of the MOMFHP$_0$ is often not solved at the root node, in contrast with what is stated in Park et al. (2017). Thus, some branching strategy must be implemented to cope with the branch and bound search. Ryan-Foster (R-F) is one of the most popular techniques for branching in set partitioning problems (see Ryan and Foster (1981)). If a fractional solution is reached at a node, R-F creates two new branches as follows: Given to elements $i_1, i_2 \in I$, they may never go together on a set in the whole branch, or they may always go together, i.e., if one of them belongs to a set S, the other one must also be included in S.

To implement this branching, we can take advantage of the w_i variables defined on the previous section for the pricing subproblem, to easily adapt this way of branching in our problem, by means of the following constraints:

A) $w_{i_1} + w_{i_2} = 1$ ensuring that elements i_1 and i_2 are not assigned to the same hyperplane.

B) $w_{i_1} = w_{i_2}$ ensuring that elements i_1 and i_2 are assigned to the same hyperplane.

Moreover, in our formulation there is a new case in which, despite the fact of having fractional solutions on a node, we will not create new branches following the R-F rule. This fact is motivated because in our problem may appear different columns (different y-variables) but being associated to the same set S, although possibly with different hyperplanes.

Let $S \subseteq I$ be a subset of points and let $y_S^1, ..., y_S^q$ be fractional variables for the same set S although with different hyperplanes $\mathcal{H}(\omega^i, \omega_0^i)$, $i = 1 \dots q$, with $q > 1$, such that $\sum_{i=1}^{q} y_S^i = 1$. If there are no more fractional variables, or the rest of the fractional variables of the node satisfy the same conditions for some other subsets of points, we cannot apply R-F rule and either the node need not be branched (see Theorem 3.2 and Remark 3.4) or a different branching strategy must be implemented in these cases.

Without loss of generality, we will describe the new branching for the case in which two fractional variables, y_S^1 and y_S^2, with hyperplanes $\mathcal{H}(\omega^1, \omega_0^1)$ and $\mathcal{H}(\omega^2, \omega_0^2)$, are obtained in a node for the same subset S. In this situation, the new branching rule that we propose creates three new branches as follows:

1. A branch where $y_S^1 = 1$ meaning that this variable will be in the solution in this branch. This is easily implemented in the pricing routine since it amounts to avoid considering the elements in S in any further column in that branch

because they are already in the set S which is part of the solution. Therefore, it suffices to fix the variables $w_i = 0,\ \forall i \in S$ in all the subproblems in the branch.

2. Analogously, it creates another branch where $y_S^2 = 1$.

3. The third branch sets $y_S^1 = y_S^2 = 0$. This branch represents the case in which none of the original fractional solutions are part of the integer solution. Once again, this can be enforced by adding the following constraints to the pricing subproblems of the branch:

$$\left(\left(|S| - \sum_{i\in S} w_i\right) + \left|\,|S| - \sum_{i=1}^{n} w_i\right| + \sum_{\ell=1}^{d} |\omega_\ell^j - \omega_\ell^*| + |\omega_0^j - \omega_0^*|\right) \cdot M \geq 1,$$

for j=1,2, and M a big enough constant, where ω^* and ω_0^* define the new hyperplane $\mathcal{H}(\omega^*, \omega_0^*)$. These constraints will make the problem infeasible if and only if all the individuals in S, and only the individuals of S, belong to the new solution, and moreover, the new solution provides a hyperplane $\mathcal{H}(\omega^*, \omega_0^*)$ that is equal to $\mathcal{H}(\omega^1, \omega_0^1)$ or $\mathcal{H}(\omega^2, \omega_0^2)$.

The alternative branching may be necessary in case of using general norm based residuals. Nevertheless, as we show below, the situation is simpler using vertical distance residuals.

Theorem 3.2. *Ryan and Foster branching is enough in the set partitioning formulation of* (MOMFHP$_0$) *for the vertical distance residuals: If for a subset of points $S \subseteq I$, there exists a fractional solution $0 < y_S < 1$ with $\#\{y_S \neq 0\} > 1$, at a node of the branch and bound tree then there exists an explicit solution that combines these variables to obtain a single one satisfying $y_S = 1$ ($\#\{y_S \neq 0\} = 1$).*

Proof. Let us consider a subset of points $S \subseteq I$. At a fractional node, we can have two possible scenarios: 1) $\#\{y_S \neq 0\} = 1$, hence, it would exist a single hyperplane (a facility) $\mathcal{H}(\omega, \omega_0)$ that would serve the points of S, and hence, R-F branching is enough, and 2) $\#\{y_S \neq 0\} > 1$. This latter case needs a further analysis since it may seem as if more than one facility would need to be involved to optimally serve the points in S.

Without loss of generality we can assume $\#\{y_S \neq 0\} = 2$ (a case with $\#\{y_S \neq 0\} > 2$ can be treated sequentially by smaller problems with two solutions). In this situation there are two variables, y_S^1 and y_S^2, with values σ and $1-\sigma$, $\sigma \in (0,1)$, so that $y_S^1 + y_S^2 = 1$. These variables are represented by two hyperplanes $\mathcal{H}(\omega^1, \omega_0^1)$ and $\mathcal{H}(\omega^2, \omega_0^2)$, where the cost of a point $i \in S$ with coordinates $x \in \mathbb{R}^p$, e_i, is given by $e_i = \sigma D(x, \mathcal{H}(\omega^1, \omega_0^1)) + (1-\sigma) D(x, \mathcal{H}(\omega^2, \omega_0^2))$. We prove that the hyperplane $\mathcal{H}(\omega^*, \omega_0^*)$ defined as

$$\mathcal{H}(\omega^*, \omega_0^*) = \left\{ z \in \mathbb{R}^p \ : \ \sigma(\omega^1 z + \omega_0^1) + (1-\sigma)(\omega^2 z + \omega_0^2) = 0 \right\},$$

satisfies that $D(x, \mathcal{H}(\omega^*, \omega_0^*)) \leq \sigma D(x, \mathcal{H}(\omega^1, \omega_0^1)) + (1-\sigma) D(x, \mathcal{H}(\omega^2, \omega_0^2))$ for vertical distance residuals, and this would mean that there exists a unique hyperplane that optimally serves all the points in S. Therefore, considering y_S^*, no further branching is required.

If we consider the normalized hyperplanes $\mathcal{H}(\omega^1, \omega_0^1)$, and $\mathcal{H}(\omega^2, \omega_0^2)$, such that $\omega_p^1 = \omega_p^2 = -1$, then $\omega_p^* = -1$, the vertical distance from x to $\mathcal{H}(\omega^*, \omega_0^*)$ is

$$D_v(x, \mathcal{H}(\omega^*, \omega_0^*)) = \left| x_p - \omega_0^* - \sum_{\ell=1}^{p-1} \omega_\ell^* x_\ell \right|.$$

Hence,

$$\begin{aligned}
D_v(x, \mathcal{H}(\omega^*, \omega_0^*)) &= \left| x_p - \omega_0^* - \sum_{\ell=1}^{p-1} \omega_\ell^* x_\ell \right| \\
&= \left| \sigma x_p + (1-\sigma) x_p - (\sigma\omega_0^1 + (1-\sigma)\omega_0^2) - \sum_{\ell=1}^{p-1} (\sigma\omega_\ell^1 + (1-\sigma)\omega_\ell^2) x_\ell \right| \\
&\leq \sigma \left| x_p - \omega_0^1 - \sum_{\ell=1}^{p-1} \omega_\ell^1 x_\ell \right| + (1-\sigma) \left| x_p - \omega_0^2 - \sum_{\ell=1}^{p-1} \omega_\ell^2 x_\ell \right| \\
&= \sigma D_v(x, \mathcal{H}(\omega^1, \omega_0^1)) + (1-\sigma) D_v(x, \mathcal{H}(\omega^2, \omega_0^2)).
\end{aligned}$$

□

Remark 3.4. *We prove that under mild conditions, R-F branching is also enough for the ℓ_1-norm based residuals.*

Without loss of generality, assume that there is a solution with two fractional variables y_S^1 and y_S^2, with values σ and $(1-\sigma)$, and corresponding hyperplanes $\mathcal{H}(\omega^1, \omega_0^1)$ and $\mathcal{H}(\omega^2, \omega_0^2)$. Let us define the set $SP = \left\{ j : |\omega_j^1| = |\omega_j^2| = 1, j = 1, \ldots, p \right\}$. Hence, if $SP \neq \emptyset$, RF-branching is enough for the ℓ_1-norm residuals.

Indeed, let $\hat{\jmath}$ be an index such that $|\omega_{\hat{\jmath}}^1| = |\omega_{\hat{\jmath}}^2| = 1$ and define

$$sign(\hat{\jmath}) = \begin{cases} 1 & \text{if } \omega_{\hat{\jmath}}^1 \cdot \omega_{\hat{\jmath}}^2 = 1 \\ -1 & \text{if } \omega_{\hat{\jmath}}^1 \cdot \omega_{\hat{\jmath}}^2 = -1. \end{cases}$$

It is clear that for any $\hat{\jmath} \in SP$, $\omega_{\hat{\jmath}}^* = \sigma\omega^1 + \text{sign}(\hat{\jmath})(1-\sigma)\omega^2$ satisfies $\|\omega_{\hat{\jmath}}^*\|_\infty = 1$. Consider for any $\hat{\jmath} \in SP$ the hyperplane

$$\mathcal{H}(\omega_{\hat{\jmath}}^*, \omega_0^*) = \left\{ z \in \mathbb{R}^p \ : \ \sigma(\omega^1 z + \omega_0^1) + \text{sign}(\hat{\jmath})(1-\sigma)(\omega^2 z + \omega_0^2) = 0 \right\},$$

then for any individual $i \in S$ with coordinates $x_i \in \mathbb{R}^p$, taking into account that

$||\omega^1||_\infty = ||\omega^2||_\infty = 1$, *we obtain that*

$$\begin{aligned} D_{\ell_1}(x_i, \mathcal{H}(\omega^*, \omega_0^*)) &= \frac{|\omega_0^* + \sum_{\ell=1}^p \omega_\ell^* x_{i\ell}|}{||\omega^*||_\infty} \\ &\leq \sigma \frac{|\omega_0^1 + \sum_{\ell=1}^p \omega_\ell^1 x_{i\ell}|}{||\omega^1||_\infty} + (1-\sigma)\frac{|\omega_0^2 + \sum_{\ell=1}^p \omega_\ell^2 x_{i\ell}|}{||\omega^2||_\infty} \\ &= \sigma D_{\ell_1}(x_i, \mathcal{H}(\omega^1, \omega_0^1)) + (1-\sigma) D_{\ell_1}(x_i, \mathcal{H}(\omega^2, \omega_0^2)). \end{aligned}$$

and hence, $y_S^* = 1$ *is an optimal solution for the problem.*

3.5 Computational results

A series of computational experiments has been performed in order to test the two proposed methodologies. We consider two different sets of instances, one based on Eilon et al. (1971) dataset and another on synthetic data. For all of them we solve (MOMFHP$_0$) for four different objective functions: Weber (W), Center (C), $\lceil \frac{n}{2} \rceil$-Centrum (K) ($\lambda = (\overbrace{1, \ldots, 1}^{\lceil \frac{n}{2} \rceil}, 0, \ldots, 0)$) and 0.9-Centdian (D) ($\lambda = (1, 0.9, \ldots, 0.9)$) and with the two proposed approaches: the compact approach based on formulation (MOMFHP) and with the branch-and-price methodology. We test the performance of the algorithms on two different types of residuals: ℓ_1-norm based residuals and absolute value vertical distance residuals.

The models were coded in C and solved with SCIP v.6.0.1 using as optimization solver CPLEX 12.8 in a Mac OS El Capitan with a Core i7 CPU clocked at 2.8 GHz and 16GB of RAM memory. A time limit of 5 hours was fixed for all the instances. It is well-known in the field of location analysis that continuous multifacility ordered median problems are very difficult to solve and already problems of moderate sizes ($n = 50$ demand points) can not often be solved to optimality (see e.g., Blanco et al. (2016)). The same or even a harder behavior should be expected here since these problems introduce a new degree of difficulty in the representation of general distance based residuals.

3.5.1 Eilon et al. (1971) dataset

First, we tested our approach on instances based on the classical planar 50-points dataset provided by Eilon et al. (1971). We randomly generate five instances from such a dataset with sizes $n \in \{20, 30, 40, 45\}$ and the entire complete original instance with $n = 50$. We run the models for $m \in \{2, 5\}$ hyperplanes. The average results obtained for these instances are shown in tables 3.1 and 3.2. There, for each combination of n (size of the instance), m (number of hyperplanes to be located) and `type` (ordered median objective function to be minimized), we provide both for

the compact formulation MOMFHP (Compact) and for the branch-and-price (B&P) approach: the CPU time in seconds needed to solve the problem (`Time` (secs.)), the MIP Gap (`GAP`) remaining after the time limit, the number of nodes (`Nodes`) explored in the branch and bound tree and the RAM memory (`Memory` (MB)) in Megabytes required during the execution process. Within each column (`Time`, `GAP`, `Nodes` and `Memory`), we highlight in bold the best result between the two formulations, namely Compact or B&P. Table 3.1 gives the results for the models with vertical distance residuals while Table 3.2 provides the results for the ℓ_1-norm residuals.

As expected, the difficulty of the problem increases with n and m. Problems with smaller n are easier and $m = 2$ is also easier than $m = 5$. We also observe in Table 3.1 that the B&P approach is more efficient than formulation MOMFHP in all cases with the only exception of center problems. For that type of problem with vertical distance residuals the compact formulation is able to solve all instances in all cases whereas the B&P reports an overall gap of 20.36%. As it can be expected the number of nodes to be explored in order to solve the problems is several orders of magnitude larger for the compact formulation than for the B&P algorithm. This fact shows that the former formulation is much less accurate than the latter thus requiring many more nodes to be explored to solve the problems, implying a better scalability of the B&P approach. In addition, MOMFHP requires very large RAM memory resources since already for $n = 50$ points, it demands, in some cases, more than 11 GB whereas B&P solves the problems using at most 4 GB of RAM memory.

Turning to Table 3.2 we observe, as expected, that using ℓ_1-norm residuals make problems harder to solve mainly due to the representation of the projections point-to-hyperplane stated in Remark 3.3. In this case, the overall gaps increase from 29.36% and 11.24% in Table 3.1, for MOMFHP and B&P, respectively, to 62.69% and 20.53%. This behavior is more severe for MOMFHP because already for $n = 20$ and $m = 5$ that formulation is not able to certify optimality for any of the problems regardless of the type within the time limit. On the contrary, B&P is affected less and its behavior is similar to what one observes for vertical distance in Table 3.1. The rest of comments regarding number of nodes and memory requirements are similar to those given previously for vertical distances.

3.5.2 Synthetic Instances

We have also randomly generated another set of instances to evaluate the performance of the two solution approaches depending on the space dimension (p). We have generated five instances of random points in the unit hypercube for each meaningful combination of $n \in \{20, 30, 40, 45, 50\}$, $m \in \{2, 5, 10\}$ and $p \in \{2, 3, 8\}$ (note that for these datasets, we have included additionally $m = 10$ to analyze how increasing the number of hyperplanes affects the complexity for larger space dimension

			Time (secs.)		GAP		Nodes		Memory (MB)	
n	m	type	Compact	B&P	Compact	B&P	Compact	B&P	Compact	B&P
20	2	W	**2.08**	68.15	0.00%	0.00%	2878	**2**	**3**	23
		K	**2.10**	69.18	0.00%	0.00%	2878	**2**	**3**	23
		D	**6.70**	86.14	0.00%	0.00%	2904	**2**	**3**	24
		C	**0.11**	3171.27	0.00%	0.00%	**35**	5070	**1**	1347
	5	W	12411.90	**23.48**	18.67%	**0.00%**	23623465	**16**	2226	**13**
		K	12422.17	**23.31**	18.70%	**0.00%**	23560539	**16**	2221	**13**
		D	13082.16	**43.96**	18.67%	**0.00%**	24131841	**26**	2161	**15**
		C	**103.70**	2798.43	0.00%	0.00%	204693	**12259**	**36**	300
Average 20:			4753.86	**785.49**	7.00%	**0.00%**	8941154	**2174**	832	**220**
30	2	W	**52.13**	1439.90	0.00%	0.00%	60401	**11**	**9**	109
		K	**52.77**	1440.09	0.00%	0.00%	60401	**11**	**9**	109
		D	**57.57**	2410.21	0.00%	0.00%	62889	**7**	**9**	107
		C	**0.17**	18000.00	**0.00%**	25.73%	**40**	2833	**3**	4033
	5	W	18000.00	**654.68**	87.13%	**0.00%**	22547601	**109**	7194	**47**
		K	18000.00	**653.84**	87.11%	**0.00%**	22459376	**109**	7161	**47**
		D	18000.00	**242.66**	83.94%	**0.00%**	22252049	**25**	7240	**41**
		C	**349.26**	11310.31	**0.00%**	28.76%	503141	**7537**	**89**	1318
Average 30:			6814.04	**4518.97**	32.27%	**6.81%**	8493237	**1330**	2714	**726**
40	2	W	**1870.93**	18000.00	**0.00%**	11.71%	1453146	**1**	**91**	251
		K	**1923.60**	18000.00	**0.00%**	11.73%	1453146	**1**	**91**	248
		D	**1765.95**	18000.00	**0.00%**	10.56%	1290038	**1**	**87**	244
		C	**0.26**	17809.35	**0.00%**	22.50%	**81**	408	**4**	1264
	5	W	18000.00	**15077.97**	99.96%	**1.88%**	15197462	**280**	9801	**141**
		K	18000.00	**15029.85**	99.96%	**1.77%**	15210786	**281**	9792	**142**
		D	18000.00	**3346.72**	99.83%	**0.00%**	14810951	**864**	9700	**183**
		C	**982.10**	12375.95	**0.00%**	18.17%	1196810	**1358**	**205**	1559
Average 40:			**7567.91**	14705.75	37.47%	**9.79%**	6326552	**399**	3722	**504**
45	2	W	**10238.75**	18000.00	**0.00%**	27.97%	6492828	**1**	**219**	351
		K	10438.84	**9514.96**	**0.00%**	5.05%	6532368	**9**	**208**	401
		D	**10192.16**	18000.00	**0.00%**	45.37%	6066819	**1**	**217**	336
		C	**0.35**	14541.26	**0.00%**	7.58%	**133**	2286	**5**	932
	5	W	18000.00	18000.00	99.86%	**29.67%**	12121664	**32**	10427	**139**
		K	18000.00	18000.00	100.00%	**47.44%**	12720196	**25**	11047	**142**
		D	18000.00	**7525.10**	99.46%	**0.01%**	12193404	**2383**	9076	**336**
		C	**1268.05**	17486.12	**0.00%**	36.94%	1570195	**1490**	**244**	2245
Average 45:			**10767.32**	15133.56	37.41%	**25.00%**	7212201	**778**	3930	**610**
50	2	W	18000.00	18000.00	22.46%	**3.48%**	9401360	**1**	1593	**582**
		K	18000.00	18000.00	22.57%	**3.48%**	9275884	**1**	1584	**583**
		D	18000.00	18000.00	21.06%	**2.84%**	8902849	**1**	1238	**515**
		C	**0.29**	18000.00	**0.00%**	19.79%	**37**	372	**6**	923
	5	W	18000.00	18000.00	100.00%	**52.33%**	10760962	**1**	11353	**127**
		K	18000.00	18000.00	100.00%	**52.33%**	10743398	**1**	11335	**126**
		D	18000.00	18000.00	100.00%	**46.21%**	9732814	**1**	9864	**134**
		C	**1778.36**	18000.00	**0.00%**	43.86%	2234747	**470**	**281**	1432
Average 50:			**13722.40**	18000.00	45.76%	**28.04%**	7631506	**106**	4657	**553**
Total Average:			**7773.24**	9224.75	29.36%	**11.24%**	7737963	**1120**	2888	**517**

Table 3.1: Results for Eilon et al. (1971) instances for vertical distance.

			Time (secs.)		GAP		Nodes		Memory (MB)	
n	m	type	Compact	B&P	Compact	B&P	Compact	B&P	Compact	B&P
20	2	W	166.73	**136.75**	0.00%	0.00%	163750	**21**	**17**	30
		K	167.49	**136.65**	0.00%	0.00%	163750	**21**	**17**	30
		D	624.85	**103.66**	0.00%	0.00%	690721	**26**	40	**30**
		C	**0.98**	10126.55	**0.00%**	5.13%	**1398**	3597	**3**	2141
	5	W	18000.00	**111.13**	100.00%	**0.00%**	29829455	**32**	10437	**14**
		K	18000.00	**109.85**	100.00%	**0.00%**	30416596	**32**	10655	**14**
		D	18000.00	**56.87**	100.00%	**0.00%**	31528234	**15**	10624	**13**
		C	18000.00	**15315.35**	100.00%	**12.23%**	40775405	**18269**	7513	**2118**
Average 20:			9120.10	**3262.10**	50.00%	**2.17%**	16696164	**2752**	4913	**549**
30	2	W	13046.35	**4509.86**	28.75%	**0.00%**	13086135	**26**	1187	**123**
		K	13034.52	**4507.75**	28.74%	**0.00%**	13098248	**26**	1188	**123**
		D	11959.18	**4595.89**	26.93%	**0.01%**	13057250	**27**	1724	**127**
		C	**2.92**	12061.62	**0.00%**	19.61%	1192	**507**	**4**	2154
	5	W	18000.00	**947.24**	98.89%	**0.00%**	20504433	**35**	10616	**39**
		K	18000.00	**927.14**	98.88%	**0.00%**	21254042	**35**	11006	**39**
		D	18000.00	**1885.54**	100.00%	**0.00%**	20197549	**140**	10972	**49**
		C	**14811.15**	18000.00	80.00%	**46.40%**	25951648	**3028**	8158	**3031**
Average 30:			13356.84	**5929.39**	57.77%	**8.25%**	15893812	**478**	5607	**711**
40	2	W	18000.00	18000.00	42.82%	**6.40%**	13047861	**1**	2218	**201**
		K	18000.00	18000.00	42.95%	**6.40%**	12998370	**1**	2214	**201**
		D	18000.00	18000.00	65.74%	**7.03%**	10642421	**1**	1809	**213**
		C	**2.64**	17184.72	**0.00%**	39.56%	3792	**124**	**5**	1593
	5	W	18000.00	18000.00	100.00%	**39.89%**	14778541	**49**	10698	**98**
		K	18000.00	18000.00	100.00%	**39.20%**	15280145	**59**	11070	**101**
		D	18000.00	18000.00	100.00%	**35.47%**	14043320	**111**	9495	**145**
		C	18000.00	18000.00	100.00%	**60.84%**	23716675	**299**	11125	**1002**
Average 40:			**15750.45**	17900.37	68.94%	**29.35%**	13063891	**81**	6079	**444**
45	2	W	18000.00	18000.00	42.39%	**4.85%**	10512338	**1**	2795	**287**
		K	18000.00	18000.00	49.79%	**25.56%**	10903011	**1**	2767	**299**
		D	18000.00	18000.00	62.64%	**24.59%**	8683785	**1**	2263	**296**
		C	**2.13**	18000.00	**0.00%**	41.79%	2251	**37**	**6**	1036
	5	W	18000.00	18000.00	100.00%	**49.90%**	11757058	**2**	10826	**97**
		K	18000.00	18000.00	100.00%	**55.13%**	12704430	**1**	11911	**98**
		D	18000.00	18000.00	100.00%	**48.34%**	11591052	**3**	9553	**95**
		C	18000.00	18000.00	100.00%	**61.79%**	22769758	**133**	11064	**728**
Average 45:			**15750.39**	18000.00	69.35%	**38.99%**	11115460	**22**	6398	**367**
50	2	W	18000.00	18000.00	96.53%	**8.22%**	7215656	**1**	3368	**371**
		K	18000.00	18000.00	96.51%	**8.22%**	7288856	**1**	3402	**371**
		D	18000.00	18000.00	96.34%	**8.21%**	6792290	**1**	2722	**338**
		C	**4.21**	18000.00	**0.00%**	47.22%	5241	**15**	**8**	680
	5	W	18000.00	18000.00	100.00%	**53.68%**	11063050	**1**	9171	**109**
		K	18000.00	18000.00	100.00%	**53.68%**	11115826	**1**	9212	**109**
		D	18000.00	18000.00	100.00%	**55.80%**	11147925	**1**	9840	**118**
		C	18000.00	18000.00	100.00%	**63.66%**	19632691	**52**	9468	**488**
Average 50:			**15750.64**	18000.00	86.17%	**37.34%**	9282692	**9**	5899	**323**
Total Average:			13601.88	**11593.62**	62.69%	**20.53%**	13958539	**794**	5756	**508**

Table 3.2: Results for Eilon et al. (1971) instances for ℓ_1-distance.

($p = 8$)). At this point, it is important to point out that several combinations of the above factors result in trivial problems, for instance for $n = 20$ and $m = 10$ there is always a solution passing through all the points and thus with zero objective value. All those cases that give rise to trivial solutions are not reported. Table 3.3 reports the results for the models with vertical distance residuals while Table 3.4 provides the results for the ℓ_1-norm residuals. We report the same information as the one provided in the previous section but this time the results do not distinguish the type of objective function but the dimension of the space. (Needless to say that all the results disaggregated also by type are available upon request.)

For this dataset the results reinforce our previous observations in that for problems with vertical distances (see Table 3.3), MOMFHP is much weaker than B&P for $m = 5, 10$ and in any dimension. In this case, however as seen in Table 3.3 there are cases where for $m = 2$ MOMFHP (see column *Compact*) is more efficient. Turning to problems with ℓ_1-norm residuals the performance is more homogeneous and B&P is more efficient than MOMFHP for all n, m and p. Once again, one observes that problems with ℓ_1-norm residuals are more difficult than with vertical residuals. The overall gaps increase from 51.49% and 29.93% in Table 3.3, for MOMFHP and B&P, respectively, to 83.78% and 37.41% in Table 3.4.

3.6 Scalability: bounding the error in aggregation procedures

This section is devoted to analyze the issue of scalability of our approach. We are aware that the methodology based on a branch and price algorithm may be computationally costly (we refer the reader to the Section 3.5 for further details). For that reason, we derive an approach that allows one to handle large data sets with appropriate error bounds.

Our approach is based on aggregating data to reduce the dimensionality of the original problem so that our branch and price approach can properly handle the problem. The important issue is that we can provide error bounds on these approximations that monotonically decrease with the quality of the aggregation. Obviously, aggregation strategies are not new since they have been already applied in some other areas although mostly from a heuristic point of view (see e.g., Current and Schilling (1987, 1990)).

Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ be a set of demands points. Aggregating X into a new set of demand points X' consists of replacing X by $X' = \{x'_1, \ldots, x'_n\}$ and to assign each point x_i in X to a point x'_i in X' (since usually the cardinality of the different elements of X' is smaller than the cardinality of X, several x_i may be assigned to the same x'_i and thus actually, some of the elements in X' coincide). A

			Time (secs.)		GAP		Nodes		Memory (MB)	
n	m	p	Compact	B&P	Compact	B&P	Compact	B&P	Compact	B&P
20	2	2	**11.00**	56.48	0.00%	0.00%	2710	**139**	**2**	49
		3	**4.39**	233.22	0.00%	0.00%	2922	**417**	**3**	137
		8	**30.88**	1506.93	0.00%	0.00%	29777	**1530**	**3**	782
	5	2	13017.91	**3930.76**	46.32%	**1.25%**	13909807	**40836**	2209	**430**
		3	18000.00	**4516.39**	100.00%	**19.56%**	19586370	**8330**	629	**29**
Average 20:			6212.84	**2048.76**	29.26%	**4.16%**	6706317	**10250**	569	**285**
30	2	2	**55.66**	2879.51	**0.00%**	1.34%	44038	**1039**	**7**	1218
		3	**60.81**	8779.12	**0.00%**	5.77%	48533	**1741**	**8**	2737
		8	**414.28**	18000.00	**0.00%**	67.19%	270055	**779**	**23**	1516
	5	2	13933.75	**5046.83**	74.14%	**11.01%**	11575613	**7989**	4058	**1274**
		3	18000.00	**12362.06**	100.00%	**18.32%**	15098323	**5033**	3187	**384**
	10	2	18000.00	**4523.03**	100.00%	**11.05%**	15536270	**10046**	1572	**298**
Average 30:			**8410.77**	8598.55	45.69%	**19.11%**	7095472	**4438**	1476	**1238**
40	2	2	**1490.88**	17404.04	**0.00%**	14.85%	903463	**805**	**56**	2186
		3	**1164.19**	18000.00	**0.00%**	18.04%	726140	**466**	**40**	1579
		8	**8455.38**	18000.00	**0.00%**	71.44%	4005359	**59**	**140**	417
	5	2	15809.48	**12850.71**	75.78%	**18.52%**	10566235	**3642**	5303	**1187**
		3	18000.00	18000.00	100.00%	**57.67%**	12378840	**705**	5061	**412**
	10	2	18000.00	**5498.89**	100.00%	**19.66%**	12196952	**1770**	2179	**159**
		3	18000.00	**13721.20**	100.00%	**43.81%**	12359358	**584**	1121	**79**
Average 40:			**11560.01**	14784.91	53.68%	**34.85%**	7590907	**1147**	1986	**860**
45	2	2	**11045.65**	18000.00	**4.56%**	25.70%	5700797	**587**	**316**	2353
		3	**8390.99**	18000.00	**0.00%**	25.64%	4494533	**235**	**127**	1205
		8	**13570.97**	18000.00	**33.32%**	67.26%	5409179	**11**	962	**324**
	5	2	16704.69	**16218.37**	75.04%	**33.86%**	9858979	**2061**	6104	**945**
		3	18000.00	18000.00	100.00%	**63.19%**	11396830	**353**	5914	**344**
		8	18000.00	18000.00	100.00%	100.00%	11094838	**1**	329	**64**
	10	2	18000.00	**6512.29**	100.00%	**20.37%**	11236807	**889**	2124	**111**
		3	18000.00	**9421.82**	100.00%	**28.47%**	10911247	**272**	1471	**71**
Average 45:			**15214.06**	15307.51	64.12%	**45.56%**	8762901	**551**	2168	**677**
50	2	2	**13500.09**	18000.00	18.38%	**6.95%**	6135466	**393**	**1132**	2361
		3	**13500.51**	18000.00	**20.60%**	30.71%	6182966	**112**	**989**	1209
		8	**13568.18**	18000.00	**45.65%**	67.16%	4649063	**2**	1243	**407**
	5	2	**16593.29**	18000.00	75.02%	**51.32%**	8984044	**801**	7563	**729**
		3	18000.00	18000.00	100.00%	**65.74%**	10380563	**157**	5969	**268**
		8	18000.00	18000.00	100.00%	100.00%	10407264	**1**	1472	**83**
	10	2	18000.00	**9490.74**	100.00%	**20.01%**	10892069	**363**	2483	**83**
		3	18000.00	18000.00	100.00%	**68.93%**	10139295	**214**	1537	**75**
Average 50:			**16145.28**	16946.05	69.96%	**51.35%**	8471341	**255**	2791	**652**
Total Average:			**11231.69**	11409.54	51.49%	**29.93%**	7735135	**3287**	1718	**773**

Table 3.3: Results for synthetic instances for vertical distance.

			Time (secs.)		GAP		Nodes		Memory (MB)	
n	m	p	Compact	B&P	Compact	B&P	Compact	B&P	Compact	B&P
20	2	2	2799.81	**1413.43**	**0.03%**	1.50%	4193628	**545**	**216**	424
		3	10649.52	**4226.37**	8.41%	**0.01%**	17136729	**617**	696	**693**
		8	18000.00	**14942.66**	100.00%	**36.68%**	32145208	**324**	488	**476**
	5	2	17977.28	**3686.10**	100.00%	**4.15%**	35717983	**5312**	9878	**922**
		3	18000.00	**4610.48**	100.00%	**16.99%**	39208660	**3325**	5832	**52**
Average 20:			13485.35	**5775.81**	61.69%	**11.87%**	25680442	**2025**	3422	**514**
30	2	2	11021.12	**10263.53**	43.06%	**4.35%**	11908005	**293**	2283	**1230**
		3	**13503.52**	15061.10	50.87%	**18.57%**	13408414	**178**	2644	**894**
		8	18000.00	18000.00	100.00%	**85.77%**	18682135	**14**	3855	**171**
	5	2	18000.00	**9923.88**	100.00%	**14.72%**	24028581	**1361**	11038	**1765**
		3	18000.00	**12882.81**	100.00%	**22.45%**	23654639	**631**	9211	**785**
	10	2	17998.01	**4745.50**	100.00%	**14.56%**	24603724	**1162**	8567	**169**
Average 30:			16087.22	**11815.89**	82.32%	**26.74%**	19380916	**607**	6266	**835**
40	2	2	**13500.52**	17798.69	63.24%	**14.27%**	8495764	**76**	2259	**1074**
		3	**13509.64**	18000.00	64.33%	**31.75%**	7710536	**25**	2929	**601**
		8	18000.00	18000.00	100.00%	**73.86%**	12839616	**2**	4272	**192**
	5	2	18000.00	18000.00	100.00%	**42.62%**	17696588	**193**	10524	**904**
		3	18000.00	18000.00	100.00%	**69.83%**	19758253	**76**	9542	**378**
	10	2	17688.37	**10787.01**	100.00%	**22.17%**	17749077	**212**	8306	**45**
		3	18000.00	18000.00	100.00%	**63.85%**	16865456	**111**	5522	**44**
Average 40:			**16671.31**	16943.82	89.65%	**45.48%**	14445041	**99**	6193	**463**
45	2	2	**13500.65**	17927.36	61.73%	**16.31%**	7295436	**26**	2189	**1101**
		3	**13507.23**	18000.00	74.40%	**25.74%**	5456543	**10**	2359	**602**
		8	18000.00	18000.00	100.00%	**69.62%**	10392241	**1**	4242	**243**
	5	2	18000.00	**16684.12**	100.00%	**45.50%**	13916126	**714**	9640	**729**
		3	18000.00	18000.00	100.00%	**68.74%**	15523670	**31**	9216	**294**
		8	18000.00	18000.00	100.00%	100.00%	13168789	**1**	1284	**57**
	10	2	18000.00	**13383.58**	100.00%	**24.11%**	14582103	**234**	8573	**61**
		3	18000.00	**16757.30**	100.00%	**60.65%**	14067756	**54**	6130	**59**
Average 45:			**16876.16**	17096.84	92.02%	**51.33%**	11800333	**134**	5454	**393**
50	2	2	**13500.40**	15300.83	71.34%	**10.99%**	6941692	**11**	3250	**950**
		3	**13512.06**	18000.00	60.70%	**13.84%**	5691897	**5**	1877	**1068**
		8	18000.00	18000.00	100.00%	**64.78%**	9864048	**1**	3915	**345**
	5	2	18000.00	**15543.12**	100.00%	**46.58%**	15611084	**492**	10326	**704**
		3	18000.00	18000.00	100.00%	**85.37%**	14771026	**9**	8255	**199**
		8	18000.00	18000.00	100.00%	100.00%	10436698	**1**	3278	**69**
	10	2	18000.00	**15309.03**	100.00%	**29.85%**	14012908	**273**	6934	**72**
		3	18000.00	18000.00	100.00%	**67.47%**	12123118	**26**	5274	**69**
Average 50:			**16876.66**	17023.10	91.50%	**52.36%**	11181559	**102**	5388	**434**
Total Average:			16038.45	**13854.81**	83.78%	**37.41%**	16590341	**569**	5435	**531**

Table 3.4: Results for synthetic instances for ℓ_1 distance.

possible choice can be substituting the set of original demand points by the centroids obtained by any of the available clustering techniques. In any case, when solving (MOMFHP$_0$) for X' instead of using X one incurs in aggregation errors.

Let $\mathbb{H}$ be the optimal arrangement of m hyperplanes for the problem and $\mathbf{e} = (e_1, \ldots, e_n)$ with $e_i = \varepsilon_{x_i}\Big(\mathbb{H}\Big)$, for $i \in I$, the residuals with respect to $\mathbb{H}$. Analogously, let $\mathbb{H}'$ be the optimal arrangement for the demand points in X' and $\mathbf{e}'$ the vector of residuals.

Theorem 3.3. *Let $T = \max\limits_{i=1,\ldots,n} \mathrm{D}(x_i, x_i')$. Then, the following relation holds:*

$$|\mathrm{OM}_\lambda(\mathbf{e}') - \mathrm{OM}_\lambda(\mathbf{e})| \leq 2\mathrm{OM}_\lambda(T, \ldots, T). \tag{3.29}$$

Proof. First of all, observe that, based on the triangular inequality, for any $\mathbb{H}$

$$\varepsilon_{x_i}(\mathbb{H}) \leq \varepsilon_{x_i'}(\mathbb{H}) + \mathrm{D}(x_i, x_i'), \forall i = 1, \ldots, n.$$

Let us also consider the vector $\mathbf{t} = (\mathrm{D}(x_1, x_1'), \ldots, \mathrm{D}(x_n, x_n'))$ of distances from the original points in X to their corresponding points in X' and denote by $\tilde{\mathbf{e}} = (\varepsilon_{x_1'}(\mathbb{H}), \ldots, \varepsilon_{x_n'}(\mathbb{H}))$. Since the function OM is non-decreasing monotone and sub-linear, it follows that:

$$\mathsf{OM}_\lambda(\mathbf{e}) \leq \mathsf{OM}_\lambda(\tilde{\mathbf{e}} + \mathbf{t}) \leq \mathsf{OM}_\lambda(\tilde{\mathbf{e}}) + \mathsf{OM}_\lambda(\mathbf{t}).$$

Hence, since $T \geq \mathrm{D}(x_i, x_i')$ for all $i = 1, \ldots, n$, we get that:

$$|\mathsf{OM}_\lambda(\mathbf{e}) - \mathsf{OM}_\lambda(\tilde{\mathbf{e}})| \leq \mathsf{OM}_\lambda(T, \ldots, T).$$

From the above inequality we can apply (Geoffrion, 1977, Theorem 5) to conclude that

$$|\mathsf{OM}_\lambda(\mathbf{e}') - \mathsf{OM}_\lambda(\mathbf{e})| \leq 2\mathsf{OM}_\lambda(T, \ldots, T).$$

□

The difference considered in the above theorem is the excess due to the implementation of an approximate solution based on the reduced model with data set X' rather than the correct optimal solution for the original data in the larger set X. This result allows us to scale our CG algorithm to problems of any size using aggregation techniques and providing estimates on the deviation from the optimal value.

We illustrate the application of the above result including the percent error obtained aggregating to 20 points some of our random problems with 50 points by the 20-mean clustering technique. As one can see in Table 3.5 the percent errors

		error (%)	
m	type	$p = 2$	$p = 3$
2	W	3.48	2.78
	K	-17.84	-16.52
	C	4.40	5.90
	D	3.59	2.87
5	W	-0.16	1.21
	K	-9.17	-2.99
	C	6.60	20.86
	D	0.16	-11.16

Table 3.5: % aggregation errors for 50 points problems and vertical distance.

are small. Observe that in some cases they are even negative, for problems that were not solved to optimality, and where the hyperplanes obtained by aggregating points, once evaluated on the actual 50 points, produce a smaller error than the upper bound found by the algorithm on the original dataset.

3.7 Conclusions

This chapter considers the problem of locating a given number of hyperplanes in order to minimize an objective function of the distances from a set of points. Each point is assigned to its closest hyperplane, thus inducing as many clusters as the number of fitting hyperplanes. The distance from each point to its corresponding fitting hyperplane can be seen as a residual and these residuals are aggregated using ordered median functions that are ordered weighted averages representing different types of utilities. Two exact approaches are presented to solve the problem. The first one is based on a compact mixed integer formulation whereas the second one is an extended set partitioning formulation with an exponential number of variables that is handled by a branch-and-price approach. To enhance the performance of this last method we have developed a generator of initial feasible solutions based on geometrical properties of the optimal solutions of the hyperplane location problem that we have also derived in this chapter, and that are used to initialize the column generation routine of this branch-and-price. We have also presented a heuristic pricing strategy that is used in combination with the exact one to speed up some pricing iterations. We report the comparison of both method to solve the problem in two different datasets on an extensive battery of computational experiments. The issue of scalability of the exact methods is also analyzed obtaining theoretical upper bounds of the error induced by some aggregated versions of the original dataset.

Chapter 4

SVM-based classification with label noise

In this chapter we propose novel methodologies to optimally construct SVM-based classifiers that takes into account that label noise occur in the training sample. We propose different alternatives based on solving Mixed Integer Linear and Non Linear models by incorporating decisions on relabeling some of the observations in the training dataset. The first method incorporates relabeling directly in the SVM model while a second family of methods combines clustering with classification at the same time, giving rise to a model that applies simultaneously similarity measures and SVM. Extensive computational experiments are reported based on a battery of standard datasets taken from UCI Machine Learning repository, showing the effectiveness of the proposed approaches.

4.1 Introduction

Amongst the most relevant applications of classification methods there are those related with security, as in spam filtering or intrusion detection. The main difference of these applications with respect to other uses of classification approaches is that malicious adversaries can adaptively manipulate their data to mislead the outcome of an automatic analysis. For instance, spammers often modify their emails by obfuscating words which typically appear in known spam or by adding words which are likely to appear in legitimate emails. Also, as stated in Weerasinghe et al. (2019), when Machine Learning algorithms utilized in safety-critical environments are compromised by adversaries, it could even result in loss of human lives. Note that, doubting on the reliability of the labels on the target variable is usual when having suspicions about the possibility of an intentional flip amongst these labels. However, it is not by far the only case in which one must think about this possibility. Nowadays, it is commonly said that data scientists spend a large percentage of their time dealing with collecting and preprocessing data, meanwhile the remainder is used to model and extract information from databases. Mistakes converted into wrong label assignments are very likely to happen. For instance, data can be wrongly identified at the very beginning of the data collection phase, or code errors can occur when preprocessing a database, leading to a dataset with label noise. Then, one has to, not only derive a classification rule from a training sample, able to adequately classify out-of-sample data, but also to take into account that some of the labels might be incorrect. The goal of this chapter is to analyze the power of using Mathematical Programming tools for the label noise detection when constructing a SVM classifier. As pointed out in Ganapathiraju et al. (2000), amongst all the available optimization-based classifiers, SVM particularly suffer the effect of noisy labels because their reliance on support vectors and the feature interdependence assumption.

Analyzing the vulnerabilities of classifiers and their robustness against attacks, to better understand how their security may be improved, has recently received growing interest from the scientific community. Bi and Zhang (2005) provided robust alternatives when the features of the training sample observations are corrupted. On the other hand, Biggio et al. (2011) proposed an algorithmic approach to handle adversarial modifications of the labels, in case the labels are independently flipped with the same probability, by correcting the kernel matrix. According to Nalepa and Kawulok (2019), three main groups of approaches for dealing with noisy datasets have been already proposed in the literature: (1) Design of algorithms which filter noisy and/or mislabeled vectors from the input data (as in Ekambaram et al. (2016), or Han and Chang (2013)); (2) Construction of robust classifiers against noisy labelling (see Duan and Wu (2016); Natarajan et al. (2017)); and (3) Use of noise models in parallel with the obtention of the classifier, which are finally coupled for a higher-quality classification (see Ganapathiraju et al. (2000); Weerasinghe et al. (2019)). Further details on the different approaches to deal with datasets containing mislabeled observations can be found in the survey in Frénay and Verleysen (2013).

Most recent methodologies to deal with noisy datasets are sequential. Thus, loosing the optimal performance obtained by one shot methods based on Mathematical Programming approaches. For instance, in the recent method presented in Northcutt et al. (2021), based in SVM with *Confident Learning (SVM-CL) approach,* the authors propose a probabilistic method in three sequential phases: 1) estimate the transition matrix of class-conditional label noise, 2) filter out noisy examples, and 3) train the dataset once noisy data are removed via Co-Teaching. Analogously, in de França and Coelho (2015) it is proposed a novel method in which first the training sample is biclustered (see e.g., (Yang et al., 2003)) trying to capture correlation between features and observations, next the training sample is modified according to the biclusters, and then the classification is performed on the modified dataset. Furthermore, there are some *globally optimal* methods that have been proposed in the literature. In particular, in Bertsimas et al. (2019), the authors present different robust adaptations of classical classification methods to deal with uncertainty in labels and/or features in the training sample.

In contrast to those methods that have been already proposed to deal with classification and noisy labels, our approach simultaneously construct a SVM-based classifier and re-label observations, leading to an optimal method. In addition, this approach allows one to get separating hyperplanes that would had been impossible to obtain throughout standard SVM and that report better results for many different problems.

Although the method proposed in Bertsimas et al. (2019) also optimally constructs the classifier under the presence of noisy labels, it is thought to be robust

against the worse possible situation. On the contrary, our method builds the hyperplanes always on the convenience of finding good classifiers and not to be protected against the worst possible flip of labels which results in better classifiers in most scenarios.

The construction of SVM-based classifiers that simultaneously relabel observations has many advantages when dealing with label noise datasets, but also when working on problems in which false positives and false negatives have different misclassifying costs. Also, in problems with unbalanced classes (as for instance in datasets on fraud with credit card transactions in which around a 99.9% of the observations are not fraudulent transactions (Maldonado et al., 2017) or in the number of claims in non-life insurances (Boucher et al., 2009)). In Figure 4.1 we illustrate this situation. One can observe in the left picture the projection on the plane of a set of observations labeled by fraudulent (red) and non fraudulent (green) transactions. Linear separators seems to be impossible to construct for this instance, but also non linear classifiers may result in overfitting. However, as shown in the right picture, if one allows a few of the labels to be changed, one can obtain better classifiers. Note that in this case, false positives are more costly than false negatives (since asking for a little more of information via text message on the phone normally solves this true negative cases). It is also important to remark that this separating hyperplane could not have been obtained through standard SVM since all the support vectors belong to the same class (green points).

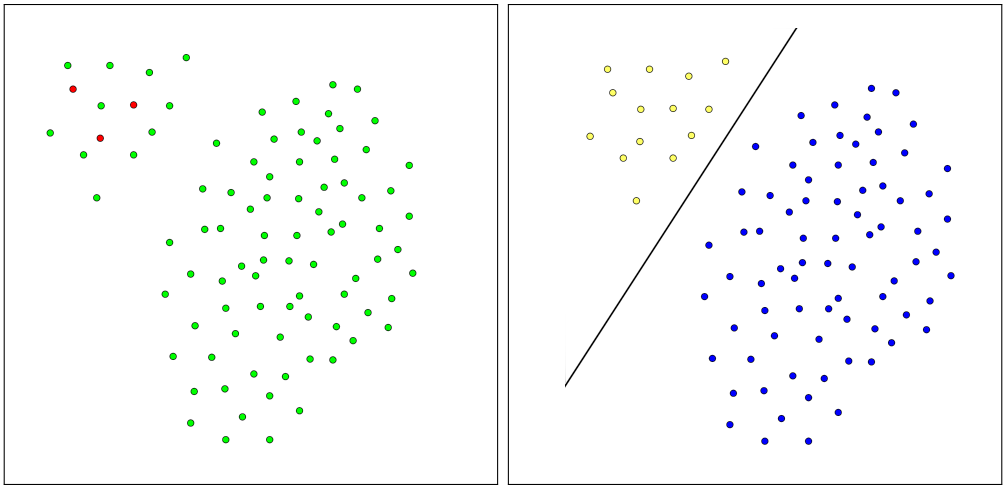


Figure 4.1: Original data (left) and optimal hyperplane separating re-labeled classes with our method (right).

In this chapter we propose two different approaches. We present a model in which re-labeling observations depends on the errors of the SVM-based method itself searching for a compromise between the gain obtained in misclassification error and

margin and the penalty paid for each change of labels. On the other hand, we will also introduce two models in which re-labeled observations will come from similarity measures on the data. Our method is distribution-free so that it does not assume any distribution on the dataset and the detection of mislabeled observations and the construction of the classifier is optimal based on solving an add-hoc Mathematical Programming problem.

To asses the validity of these methods we have performed a battery of computational experiments on 7 different real datasets. For these datasets we have repeated the experiments for 5 different scenarios, by randomly flipping a 0%, 20%, 30%, 40% or 50% of the labels in the original data. When comparing our method with respect to classical SVM, and with SVM-CL from (Northcutt et al., 2021), we can see that ours gets better results on noisy label datasets.

4.2 Mathematical Programming formulations

4.2.1 Preliminaries

In this chapter we consider a given training sample $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times \{-1, +1\}$ and a SVM-based separating hyperplane $\mathcal{H} = \{z \in \mathbb{R}^p \ : \ \omega' z + \omega_0 = 0\}$, $\omega \in \mathbb{R}^p$, $\omega_0 \in \mathbb{R}$.

We show in Figure 4.2 an example where we can see a set of points belonging to two different, blue and green, classes (left picture) and its SVM optimal solution for a given parameter c (right picture). The black line is the separating hyperplane while the other two parallel lines are delimiting the strip, $\mathcal{S}$, between classes. The points that lie on these parallel lines, the boundary of the strip, are the so called support vectors, and they verify that $|\omega' x_i + \omega_0| = 1$. Finally, we represent in red color the magnitude of the errors induced by margin violations.

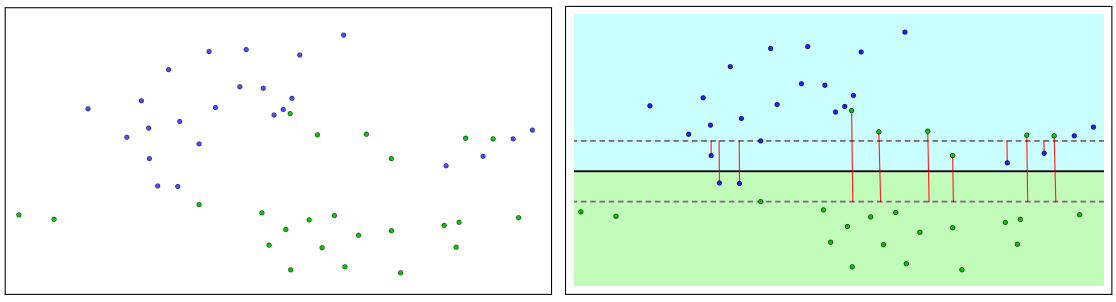


Figure 4.2: Original set of points (left) and optimal SVM solution on these points (right).

If we further analyze the above dataset, we can see that there are four blue observations at the very right of the dataset, and two green observations on the left

that have a strong impact when building the classifier. These observations do not allow one to construct a SVM separator of the dataset as the one we can see in Figure 4.3, since that would lead to very big misclassification errors with a very tiny margin.

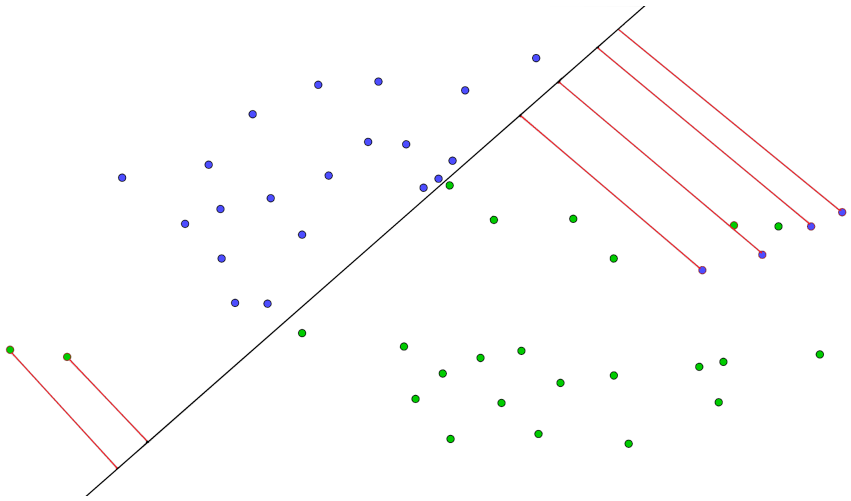


Figure 4.3: Not optimal solution on the SVM problem.

Moreover, there are another two green observations, besides the two on the left, that are closer to the blue cloud of points than to the green one. Hence, if we could consider that these four green points and the four blue ones on the right were wrongly labeled (because of their closeness to the rest of points), we might consider a separating hyperplane with a slope like the one presented on the left of Figure 4.4 as a better classifier. However, this separating hyperplane would be impossible to obtain with the SVM model since all the support vectors belong to the same class and to avoid huge misclassification errors the model would forbid such a slope.

Motivated by the above kind of configurations, we have studied different models in which a separating hyperplane is obtained not only based on the original labels but also on the possibility of relabeling some of the original observations of the training sample at a given penalty cost. We say that an observation is relabelled if one of the following assumptions occurs:

$$y_i = \pm 1 \text{ but our model considers that } y_i = \mp 1.$$

We will use the notation $\hat{y}_i$ to represent the class that the model is considering for observation i. Hence, an observation is said to be relabelled if $y_i \neq \hat{y}_i$.

Following the example shown in Figures 4.2 and 4.3, we can see on the right of Figure 4.4 the solution of our model, with a separating hyperplane with the desired

slope. Considering the original classes (blue and green), purple points represent the points that the model considers to be blue (despite of their actual label), and orange points represent the points that the model considers to be green. This separating hyperplane is optimal in our problem, the model considers that support points belong to different classes (even thought that is not true regarding to the original values) and no misclassification errors appear in the solution (which is also not true for the original labels). The underlying idea in these models is that based on the geometry

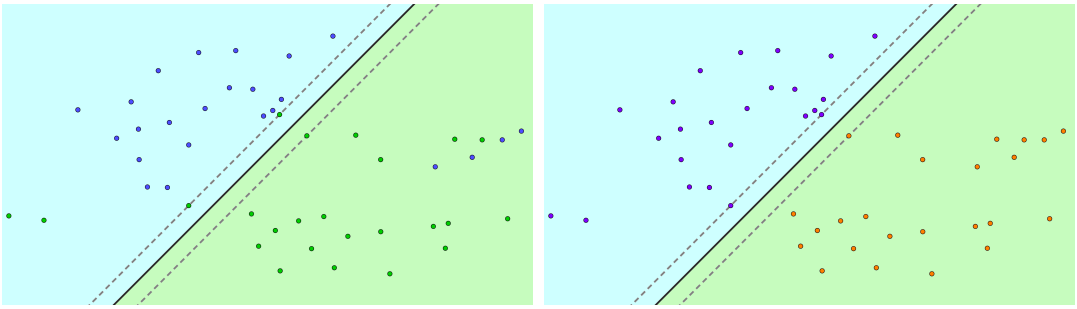


Figure 4.4: Optimal solution after re-labeling.

of the problem, relabeling some observations can lead to more robust/accurate classifiers. These classifiers can be very useful when dealing with datasets with outliers, and also in datasets in which some noise is known to be added to the data labels.

In the following we present the three mathematical optimization models that we propose to solve the problem consisting on building a hyperplane for binary classification, and, simultaneously, relabeling potential noisy observations. In the first model, relabeling labels on the original observations will be based on the errors with respect to the separating hyperplane. On the other hand, besides considering the errors with respect to the separating hyperplane, the other two models will also take into account information from data based on the geometry of the points through the k-means and the k-medians methods. Nevertheless, despite the fact that some observations are relabelled in our models, in order to make predictions, we will maintain the state for predictions on out of sample data which establishes that observations that lie on the positive half-space of the separating hyperplane will be predicted as positive class observations, meanwhile observations that lie on the negative half-space will be predicted as negative class observations.

4.2.2 Model 1: Re-label SVM

The first model that we propose relies on a very basic idea, observations will be relabelled based on the error with respect to the separating hyperplane, i.e., a penalty for each relabeling will be considered and the model will determine whether the

cost compensates the global misclassification error. Let $\hat{y}_i$ be the final label for the observation i (after relabeling), for all $i = 1, \ldots, n$. Hence, using the notation introduced before, the model can be synthetically summarized in the following way.

$$\begin{aligned}
&\min \frac{1}{2}\|\omega\|_2^2 + c_1 \sum_{i=1}^{n} d_i + \text{relabelingCost}(\hat{y}) \\
&\text{s.t.} \quad \hat{y}_i(\omega' x_i + \omega_0) \geq 1 - d_i && \forall i = 1, \ldots, n, \\
&\qquad \omega \in \mathbb{R}^p,\ \omega_0 \in \mathbb{R}, \\
&\qquad d_i \in \mathbb{R}_+, && \forall i = 1, \ldots, n, \\
&\qquad \hat{y}_i \in \{-1, 1\}, && \forall i = 1, \ldots, n.
\end{aligned}$$

The model above is a SVM model in which observations can be relabelled, and thus, instead of considering y_i on the separability constraint, the relabelled observations $\hat{y}_i$ are used. In what follows we describe how to incorporate the relabeling to the constraints and the objective function. Observe that if no cost is assumed for relabeling, the model will relabel most of the observations to obtain a null misclassification error, resulting in senseless classifiers. Thus, we model this cost with a penalty, so that the model will try to maintain the original labels on data and it will only relabel observations when a strong gain on the margin or a strong minimization on the errors is produced.

In order to derive a suitable Mathematical Programming formulation for the problem, we consider the following set of binary variables to model relabeling:

$$\xi_i = \begin{cases} 1, & \textit{if } \hat{y}_i = -y_i, \\ 0, & \textit{otherwise.} \end{cases} \quad \text{for } i = 1, \ldots, n.$$

With these variables, $\text{relabelingCost}(\hat{y}) = c_2 \sum_{i=1}^{n} \xi_i$, where c_2 is the unitary cost of relabeling. Also, to construct the classifier, we consider the following auxiliary set of continuous variables:

$$\beta_{ij} = \begin{cases} \omega_j, & \textit{if observation } i \textit{ is relabelled,} \\ 0, & \textit{otherwise.} \end{cases} \in \mathbb{R} \text{ for } i = 1, \ldots, n, \text{ for } j = 0, \ldots, p.$$

where $\beta_i = (\beta_{i1}, \ldots, \beta_{ip}) \in \mathbb{R}^p$.

Observe that, with the above notation,

$$\hat{y}_i(\omega' x_i + \omega_0) = y_i(\omega' x_i + \omega_0) - 2y_i(\beta_i x_i' + \beta_{i0})$$

Based on the discussion above, our problem can be formulated as follows:

$$\min \frac{1}{2}\|\omega\|_2^2 + c_1 \sum_{i=1}^{n} d_i + c_2 \sum_{i=1}^{n} \xi_i \qquad (\mathrm{RE}-\mathrm{SVM})$$

$$\text{s.t.} \quad y_i(\omega' x_i + \omega_0) - 2y_i(\beta_i' x_i + \beta_{i0}) \geq 1 - d_i, \qquad \forall i = 1,\dots,n, \quad (4.1)$$

$$\beta_{ij} = \xi_i \omega_j, \qquad \forall i = 1,\dots,n, j = 0,\dots,p, \quad (4.2)$$

$$\omega \in \mathbb{R}^p,\ \omega_0 \in \mathbb{R}, \quad (4.3)$$

$$\beta_i \in \mathbb{R}^p,\ \beta_{i0} \in \mathbb{R}, \qquad \forall i = 1,\dots,n, \quad (4.4)$$

$$d_i \in \mathbb{R}_+,\ \xi_i \in \{0,1\}, \qquad \forall i = 1,\dots,n. \quad (4.5)$$

In the formulation above, constraints (4.1) and (4.2) allow to model the relabelled observations whereas (4.3) declares that the coefficients of the hyperplane are continuous variables. Constraint (4.4) defines a set of variables that will be equal to the coefficients of the hyperplane when an observation is relabelled, and zero otherwise. With these new coefficients, if an observation is not relabelled, constraints (4.1) coincide with those of the classical SVM, that together with the objective function and (4.5) allow one modeling the misclassification errors as hinge losses, i.e., $d_i = \max\{0, 1 - y_i(\omega' x_i + \omega_0)\}$ for all $i = 1,\dots,n$.

Note that $(\mathrm{RE}-\mathrm{SVM})$ is a Mixed Integer Nonlinear Problem due to its objective function, because even though constraints (4.2) are written in a nonlinear way, they can be linearized as follows:

$$\omega_j - M(1-\xi_i) \leq \beta_{ij} \leq \omega_j + M(1-\xi_i), \qquad \forall i = 1,\dots,n, j = 0,\dots,p,$$
$$-M\xi_i \leq \beta_{ij} \leq M\xi_i, \qquad \forall i = 1,\dots,n, j = 0,\dots,p.$$

for $M \gg 0$ a big enough constant. Observe that one can always assume that the coefficients of the hyperplane are normalized and that $\|(\omega, \omega_0)\|_\infty \leq 1$, and then, the value of M can be fixed to one.

With the above considerations, $(\mathrm{RE}-\mathrm{SVM})$ can be reformulated as a Quadratic Mixed Integer Programming problem with linear constraints (MIQP), which can be solved by the available off-the-shelf solvers (Gurobi, CPLEX, XPRESS, ...), which use a non-linear branch and bound approach (Gupta and Ravindran, 1985) whose continuous subproblems are efficiently solved using interior-point algorithms.

4.2.3 Cluster-SVM models

The second family of models that we propose for detecting label noise in the data are based on using similarity measures on the observations. These models will be called *Cluster-SVM methods* since they perform, simultaneously, two tasks: clustering and

classification by SVM. On the one hand, the *cluster* phase of these methods will induce relabeling based on heterogeneity of the information, whereas the SVM phase computes the classifier after relabeling. We present here two different alternatives for clustering data into two groups and its linkage to a classification system: the 2-median and the 2-mean problems.

The goal of these methods is to find two clusters for a given set of observations, considering that an observation will belong to exactly one cluster. These clusters are built by finding two *distinguished points (centroids or medians)* representing each of the two groups determined by the observations closer to them, in a way that the overall sum of distances from points to their respective distinguished points is minimum. We distinguish two models under these settings by using two different distance measures: the ℓ_1 and the ℓ_2 norms.

Let us denote by $K_+ \in \mathbb{R}^p$ and $K_- \in \mathbb{R}^p$ the two (unknown) distinguished points, and $r_i = \min\{\|x_i - K_+\|, \|x_i - K_-\|\}$, the distance from the observation i to its closest distinguished points, for $i = 1, \ldots, n$ (here $\|\cdot\|$ will represent either the ℓ_1 or the ℓ_2-norm). The representation of such a closest distance to the distinguished points will be incorporated to the Mathematical Programming model using the following set of binary variables:

$$\theta_i = \begin{cases} 1, & \textit{if observation } i \textit{ is assigned to cluster } +, \\ 0, & \textit{if observation } i \textit{ is assigned to cluster } -, \end{cases} \quad \text{for } i = 1, \ldots, n.$$

These clusters represent *similar* observations and will help the SVM methodology, together with the relabeling, to find more accurate classifiers.

Combining the ideas presented on RE-SVM with the clustering based methods, we can derive a new family of models, that assign observations to two groups based on the clusters obtained by minimizing the overall sum of the norm-based distances from the data points to their corresponding reference points. Moreover, it also tries to separate as much as possible these two clusters by means of a hyperplane. Each one of the clusters is assigned to one of the differentiated classes in our classification problem. Finally, this hyperplane will induce a subdivision of the data space in a way that the decision rule of the classification problem for out-of-sample data is the same that is used in standard SVM. We present below a MIP formulation for this problem. Let $M_1, M_2, M_3 \gg 0$ be big enough positive constants and $\|\cdot\|$ representing either the ℓ_1 or the ℓ_2-norm.

$$\min \ \frac{1}{2}\|\omega\| + c_1 \sum_{i=1}^{n} d_i + c_2 \sum_{i=1}^{n} \xi_i + c_3 \sum_{i=1}^{n} r_i \qquad (\text{Cluster} - \text{SVM})$$

$$\text{s.t.} \quad y_i(\omega' x_i + \omega_0) \geq -M_1 \xi_i, \qquad \forall i = 1, \ldots, n, \qquad (4.6)$$

$$r_i \geq \|x_i - K_+\| - M_2(1 - \theta_i), \qquad \forall i = 1, \ldots, n, \qquad (4.7)$$

$$r_i \geq \|x_i - K_-\| - M_2 \theta_i, \qquad \forall i = 1, \ldots, n, \qquad (4.8)$$

$$\omega' x_i + \omega_0 \geq 1 - d_i - M_3(1 - \theta_i), \qquad \forall i = 1, \ldots, n, \qquad (4.9)$$

$$\omega' x_i + \omega_0 \leq -1 + d_i + M_3 \theta_i, \qquad \forall i = 1, \ldots, n, \qquad (4.10)$$

$$\theta_i, \ \xi_i \in \{0, 1\}, \qquad \forall i = 1, \ldots, n, \qquad (4.11)$$

$$e_i, d_i \in \mathbb{R}_+, \qquad \forall i = 1, \ldots, n, \qquad (4.12)$$

$$K_+, K_- \in \mathbb{R}^p, \qquad (4.13)$$

$$\omega \in \mathbb{R}^p, \ \omega_0 \in \mathbb{R}. \qquad (4.14)$$

Note that the constants M_1, M_2, M_3 in the formulation above must be chosen such that $M_1 > \max_{i=1,\ldots,n} \left\{ y_i \left(\sum_{j=1}^{p} x_{ij} + 1 \right) \right\}$ (considering w.l.o.g. that the coefficients are taken so that $\|(\omega, \omega_0)\|_\infty \leq 1$), $M_2 > \max\{\|x_i - x_j\| : i, j = 1, \ldots, n\}$ and $M_3 > \sum_{j=1}^{p} x_{ij} + 2 + \max\{\|x_i - x_j\|_2 : i, j = 1, \ldots, n\}$.

The objective function of Cluster − SVM aggregates the following four elements to be simultaneously optimized:

- The margin (measured with the ℓ_1 or ℓ_2 norm) has to be maximized.
- The errors of classification with respect to the separating hyperplane have to be minimized.
- Relabelled observations have to be penalized.
- Distances from observations to their reference points have to be minimized.

The aggregation of these four terms leads to define a hyperplane with a good margin, separating two *homogenous* clusters with respect to distances and classes. Constraint (4.6) enforces the positive (resp. negative) class observations to be located on the positive (resp. negative) half-space of the separating hyperplane. Each relabelled observation is penalized by c_2 units, not allowing a large number of relabeling unless it compensates large misclassification errors or unless they lead to a margin gain. This methodology allows us to keep the same decision rule for out-of-sample data as the one used in standard SVM. Constraints (4.7) and (4.8) permit to determine the closest centroid to each observation, whereas constraints (4.9) and

(4.10) enforce the misclassification errors to be computed with respect to the cluster, i.e. the classification is performed with respect to the classes $\hat{y}_i$ that have been created based on the similarity of the observations.

The above model results in two different problems depending on the norm-based distances applied.

2Median SVM Model This model results from (Cluster − SVM) using the norm ℓ_1. It will be referred to as the 2-Median SVM model. The problem turns out to be a mixed integer linear problem and can be solved using any of the off-the-shelf MIP solvers.

2Mean SVM Model This is the version of model (Cluster − SVM) using the ℓ_2. Since we are using a nonlinear norm, the 2-Means SVM results in a Mixed Integer Nonlinear Programming problem, that can be reformulated as a Mixed Integer Second Order Cone Optimization (MISOCO) problem. As for the MIP there are nowadays available off-the-shelf commercial optimization solvers implementing routines for its efficient solution.

Remark 4.1 (2-ℓ_τ Cluster SVM Model)**.** *One could also consider different ℓ_τ-norms ($\tau \geq 1$) for both the margin measure and the clusters similarity measures. In this case, the problem becomes also a MINLP problem, but based on the results provided by Blanco et al. (2014), it can be cast as a second order cone programming problem.*

4.3 Experiments

In this section we report the results of our computational experience. We have studied seven real datasets from UCI Machine Learning Repository (Lichman et al. (2013)), all of them are binary classification problems that come from different topics. The datasets used are: Statlog - Australian Credit Approval (Australian), Breast Cancer (BreastCancer), Statlog - Heart (Heart), Parkinson Dataset with replicated acoustic features (Parkinson), QSAR biodegradation (QSARbiodeg), Vertebral Column (Vertebral) and Wholesale Customers (Wholesale). The dimensions (n: number of observations, p: number of features) of these datasets is reported in Table 4.1.

For each of these datasets we have performed five different experiments. The goal in these experiments is to make predictions as accurate as possible on out of sample data. The first experiment consists on making predictions by training the models with the original data. On the other hand, in order to represent attacks in the training data, we have considered four different scenarios in which a random amount of labels, within the set $\{20\%, 30\%, 40\%, 50\%\}$, have been flipped for training data, i.e., four scenarios in which we have added some label-noise on training

data.

We have performed a 5-fold cross validation scheme. Thus, data have been split into 5 train-test random partitions. In each of these folds we have trained our models and we have used the other four folds for testing. Moreover, we have repeated this 5-fold cross validation 5 times for each dataset, in order to avoid beneficial starting partitions, and we report the average results obtained. For all the instances we have trained our three models and we have compared them with standard SVM and SVM-CL (Northcutt et al., 2021). We have considered standard SVM as benchmark since, despite the good results provided by SVM-CL for some experiments, standard SVM provided a better performance on average among all the experiments (see Table 4.1 and Figure 4.5). The measure used to evaluate the performance of the models have been the accuracy, in percentage, on out of sample data.

The parameters that appears in the different methods that we compare are validated as usual, that is, for each of the instances we perform a grid search on the cost parameters and the best result obtained in the validation sample among these parameters is the one reported. More specifically, the grids used in the experiments are the following:

SVM: $c \in \{10^i : i = -5, \ldots, 5\}$.

RE-SVM: $c_1, c_2 \in \{10^i : i = -5, \ldots, 5\}$.

2-medians-SVM: $c_1, c_2 \in \{10^i : i = -5, \ldots, 5\}$, $c_3 \in \{10^i : i = -3, \ldots, 0\}$.

2-means-SVM: $c_1, c_2 \in \{10^i : i = -5, \ldots, 5\}$, $c_3 \in \{10^i : i = -3, \ldots, 0\}$.

SVM-CL: Default tuning parameters (see (Northcutt et al., 2021)).

The Mathematical Programming models were coded in Python 3.6, and solved using Gurobi 7.5.2 on a PC Intel Core i7-7700 processor at 2.81 GHz and 16GB of RAM. Due to the complexity of the 2-means-SVM, we have helped the solver uploading an initial feasible solution that was obtained in the 2-medians-SVM problem. We have not solved to optimality all the instances, especially those with the 2-means-SVM in which the problem becomes nonlinear, and hence we have established a time limit of 30 seconds for all the experiments. This training time has sufficed to obtain rather good classifiers. Indeed, as one can observe from the results obtained, this time limit is adequate to construct robust classifiers under noisy labels. Note that not guarantying the optimality of the solutions of our models does not necessarily imply that the classifiers are not adequate.

In Table 4.1 we report the average accuracy results obtained in all the experiments for the different models and the different levels of label-noise. In such a table we have used the yellow-green color to indicate the results in which we are a $3\% - 5\%$

better than the benchmark, the green color to indicate whether we are a $5\% - 10\%$ better than the benchmark, and the cyan color to highlight the results in wich we are at least a 10% above the benchmark. Also, we show in Figure 4.5 the accuracy boxplots of the 625 instances per dataset (5 partitions $\times$ 5 scenarios $\times$ 5 folds $\times$ 5 models).

Regarding to the results, several conclusions can be pointed out:

- Our three models perform consistently better than classical SVM when the training dataset is corrupted. Besides, the stronger the percentage of flipped labels, the bigger the difference between our models' results and SVM's results. In Figure 4.5 one can check how SVM model has lower tails and wider boxes than RE-SVM.

- 2-medians-SVM and 2-means-SVM perform better than RE-SVM for heavy attacks ($40\% - 50\%$ of flipped observations). In contrast, the cluster-based models require more time to be trained than RE-SVM, both because the problems are harder to solve (apart from relabeling, the distances to the centroids and the assignments observations-to-centroids are modeled) and the number of parameters that must be tuned. In Figure 4.5 one can easily check that RE-SVM has wider boxes than 2-medians-SVM and 2-means-SVM, which are explained by the behavior of these models against the attacks.

- Our models have a better performance than the rest of approaches even for the original datasets in which no labels are flipped. This is due to the flexibility offered by methodologies, because some of the observations are allowed to be relabeled looking for a better classifier. The original datasets may contain outliers that contaminate the sample and so they deteriorate the classifier. This situation is automatically detected and fixed by our methods, by adequately relabeling observations.

- Our methods outperform SVM-CL, which is a specialized method, designed to detect noisy labels. The rationale under these results is that SVM-CL seems to wrongly identify the right distributions of the data. These mistakes propagate to the construction of the classifier since it is built on some incomplete data. This fact also results in worse accuracies than standard SVM that works with the entire dataset without paying attention to the existence of outliers.

Overall, as one may expect and it is confirmed in our computational experiments, it is better to construct the classifier without identifying *incorrectly* the noise labels (as SVM does) than using inadequate flips to build the classifier (as SVM-CL seems to do in the tested datasets). Obviously, the results in the experiments also show

that it is rather advantageous the correct identification of the wrong labels since it improves significantly the classification rates.

		Percentage of Flipped Labels				
Dataset	Method	0%	20%	30%	40%	50%
Australian (690,14)	SVM-CL	84.55	82.10	71.12	58.93	49.68
	SVM	86.11	85.43	79.23	68.13	59.47
	RE-SVM	86.42	85.68	83.37	76.97	66.13
	2-medians-SVM	86.08	85.84	84.67	78.95	69.54
	2-means-SVM	85.97	85.74	82.65	77.14	67.70
BreastCancer (683,9)	SVM-CL	95.73	91.87	87.37	78.49	58.36
	SVM	96.49	93.47	89.96	85.94	68.16
	RE-SVM	96.88	96.20	94.97	90.36	77.00
	2-medians-SVM	96.63	95.31	94.46	91.10	87.31
	2-means-SVM	96.96	95.93	95.39	93.11	90.01
Heart (270,13)	SVM-CL	78.70	71.03	60.09	56.01	49.66
	SVM	82.23	76.86	69.68	63.79	56.90
	RE-SVM	82.84	78.38	73.16	68.86	61.25
	2-medians-SVM	82.01	78.75	77.29	75.38	71.99
	2-means-SVM	82.06	78.81	77.40	75.97	72.90
Parkinson (240,40)	SVM-CL	78.18	65.56	59.47	55.58	49.29
	SVM	81.66	74.74	70.17	62.28	57.82
	RE-SVM	82.43	77.64	73.22	67.29	62.97
	2-medians-SVM	80.32	78.62	78.12	77.51	76.28
	2-means-SVM	80.47	79.22	78.78	78.20	77.03
QSARbiodeg (1055,40)	SVM-CL	81.62	78.86	74.07	56.78	46.78
	SVM	82.12	78.07	74.09	63.38	48.97
	RE-SVM	84.53	79.61	75.00	66.42	54.58
	2-medians-SVM	84.08	78.79	74.32	67.87	67.02
	2-means-SVM	83.61	78.55	74.42	67.86	66.81
Vertebral (310,6)	SVM-CL	80.94	72.79	68.54	60.53	50.69
	SVM	84.51	75.43	71.34	66.78	57.47
	RE-SVM	85.10	79.61	74.83	72.33	67.92
	2-medians-SVM	85.31	82.62	80.80	78.30	76.31
	2-means-SVM	86.28	84.32	81.77	79.91	76.76
Wholesale (440,7)	SVM-CL	88.98	85.40	78.03	57.19	45.42
	SVM	90.08	85.30	79.74	72.23	57.73
	RE-SVM	90.39	88.77	85.97	80.12	69.07
	2-medians-SVM	90.58	89.54	87.79	82.78	73.54
	2-means-SVM	91.23	89.56	87.39	85.88	82.92

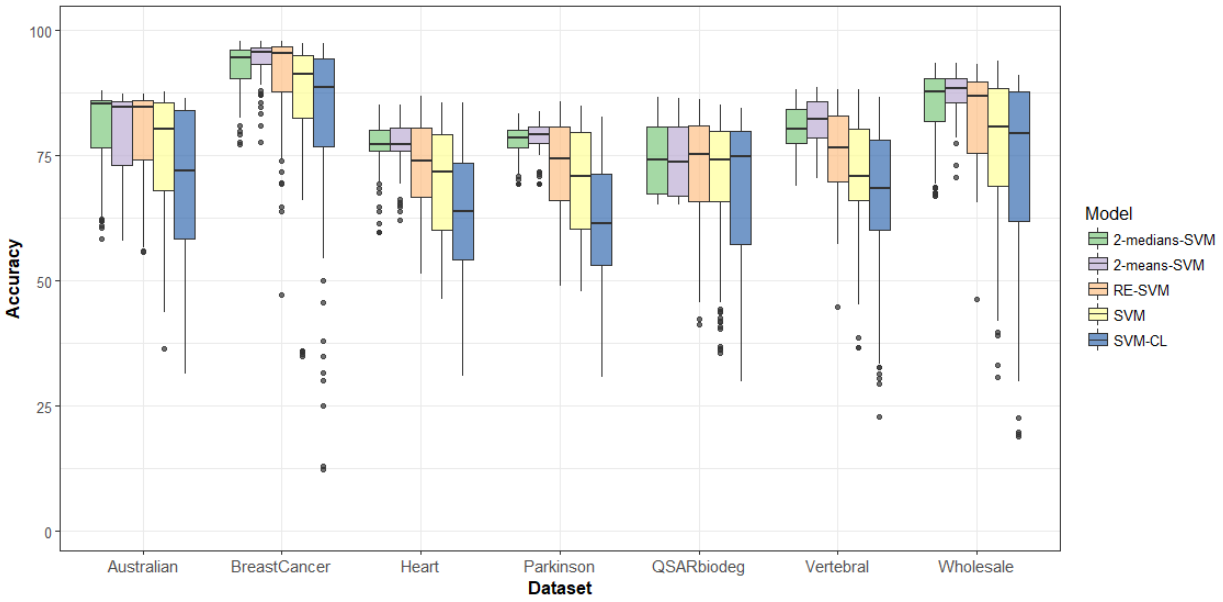


Figure 4.5: Accuracy Boxplots of the obtained accuracies.

4.4 Conclusions

This chapter presents a methodology to construct a classification rule that at the same time incorporates the detection of label noise in the datasets. Our methodology combines the power of SVM and the features of clustering analysis to simultaneously identify wrong labels to build a separating hyperplane maximizing the margin, minimizing the misclassification errors and penalizing relabeling. The rationale is simple: observations identified as wrongly labeled will be relabelled only if the gain in margin or the decrease in misclassification error compensate the flipping. In spite of its theoretical simplicity we show the exceptional performance of our methodology in a number of databases taken from the UCI repository.

These models are implemented using Mathematical Programming formulations with some integer variables (MIP). In all cases, they give rise to models that are simple and that enjoy the *quality* of being solvable by nowadays off-the-shelf commercial solvers (Gurobi, CPLEX, XPRESS...)

Our findings are not only of theoretical interest. Its practical performance when applied to databases is remarkable. In all tested cases, our methods are superior to the considered benchmark that in our case is standard SVM. Thus, they are directly applicable to datasets in which flipped labels are suspected, resulting in robust classifiers to noisy labels.

Chapter 5

Robust Optimal Classification Trees under Noisy Labels

In this chapter we propose a novel methodology to construct Optimal Classification Trees that takes into account that noisy labels may occur in the training sample. The motivation of this new methodology is based on the superaditive effect of combining together margin based classifiers and outlier detection techniques. Our approach rests on two main elements: (1) the splitting rules for the classification trees are designed to maximize the separation margin between classes applying the paradigm of SVM; and (2) some of the labels of the training sample are allowed to be changed during the construction of the tree trying to detect the label noise. Both features are considered and integrated together to design the resulting *Optimal* Classification Tree. We present a Mixed Integer Non Linear Programming formulation for the problem, suitable to be solved using any of the available off-the-shelf solvers. The model is analyzed and tested on a battery of standard datasets taken from UCI Machine Learning repository, showing the effectiveness of our approach. Our computational results show that in most cases the new methodology outperforms both in accuracy and AUC the results of the benchmarks provided by OCT and OCT-H.

5.1 Introduction

Interpretability is a crucial requisite demanded to Machine Learning methods provoked by the tremendous amount of methodologies that have arised in the last decade (Du et al., 2019). It is expected that the model that results when applying a Machine Learning methodology using a training sample, apart from being able to adequately predict the behaviour of out-of-sample observations, can be interpreted. Different tools have been applied to derive *interpretable* Machine Learning methods. One of the most popular strategies to simplify the obtained models is *feature selection*, in which a reduced set of attributes is to be selected without loosing quality in the predictions. Reducing the number of parameters to analyze, the models can be easier to understand, yielding higher descriptive accuracy. One could also consider models that can be modulated, in the sense that a great proportion of its prediction-making process can be interpreted independently. This is the case of generalized linear models (Hastie and Tibshirani, 2017). Other methods incorporate interpretability as a synonym of being able to be reproduced by humans in its entire construction(Letham et al., 2015). This is the case of Decision Trees with small depth which can be visualized and interpreted easily by users even not familiar with the tools behind their construction.

CART is the most popular Decision Trees method. In CART, one constructs the decision rule based on a hierarchical relation amongst a set of nodes which is used to define paths that lead observations from the root node (highest node in the hierar-

chical relation), to some of the leaves in which a class is assigned to the data. These paths are obtained according to different optimization criteria over the predictor variables of the training sample. The decision rule comes up naturally, the classes predicted for new observations are the ones assigned to the terminal nodes in which observations fall in. Historically, CART is obtained heuristically through a greedy approach, in which each level of the tree is sequentially constructed: starting at the root node and using the whole training sample, the method minimizes an impurity measure function obtaining as a result a split that divides the sample into two disjoint sets which determine the two descendant nodes. This process is repeated until a given termination criterion is reached (minimum number of observations belonging to a leaf, maximum depth of the tree, or minimum percentage of observations of the same class on a leaf, amongst others). In this approach, the tree grows following a top-down greedy approach, an idea that is also shared in other popular decision tree methods like C4.5 (Salzberg, 1994) or ID3 (Quinlan, 1996). The advantage of these methods is that the decision rule can be obtained rather quickly even for large training samples, since the whole process relies on solving manageable problems at each node. Furthermore, these rules are interpretable since the splits only take into account information about lower or upper bounds on a single feature. Nevertheless, there are some remarkable disadvantages in these heuristic methodologies. The first one is that they may not obtain the *optimal* classification tree, since they look for the best split locally at each node, not taking into account the splits that will come afterwards. Thus, these local branches may not capture the proper structure of the data, leading to misclassification errors in out-of-sample observations. The second one is that, specially under some termination criteria, the solutions provided by these methods can result into very deep (complex) trees, resulting in overfitting and, at times, loosing interpretability of the classification rule. This difficulty is usually overcome by pruning the tree as it is being constructed by comparing the gain on the impurity measure reduction with respect to the complexity cost of the tree.

Recently, Bertsimas and Dunn (2017) introduced the notion of OCT by approaching CART under optimization lens, providing a Mixed Integer Linear Programming formulation to optimally construct Classification Trees. In this formulation, binary variables are introduced to model the different decisions to be taken in the construction of the trees: deciding whether a split is applied and if an observation belongs to a terminal node. Moreover, the authors proved that this model can be solved for reasonable size datasets, and equally important, that for many different real datasets, significant improvements in accuracy with respect to CART can be obtained. In contrast to the standard CART approach, OCT builds the tree by solving a single optimization problem taking into account (in the objective function)

the complexity of the tree, avoiding post pruning processes. Moreover, every split is directly applied in order to minimize the misclassification errors on the terminal nodes, and hence, OCT are more likely to capture the essence of the data. Furthermore, OCT can be easily adapted in the so-called OCT-H model to decide on splits based on hyperplanes (oblique) instead of on single variables. Another remarkable advantage of using optimization tools in supervised classification methods is that features such as sparsity or robustness, can be incorporated to the models by means of binary variables and constraints Günlük et al. (2021).

At this point, we would like to finish this discussion pointing out one of the main differences between SVM, which has been deeply analyzed throughout this dissertation, and Classification Trees: SVM accounts for misclassification errors based on distances (to the separating hyperplane), i.e., the closer to the correct side of the separating hyperplane, the better, whereas in Classification Trees all misclassified observations are equally penalized.

In previous chapter we proposed different SVM-based methods that provide robust classifiers under the hypothesis of label noises. The main idea supporting those methods is that labels are not reliable, and in the process of building classification rules it may be beneficial to *flip* some of the labels of the training sample to obtain more accurate classifiers. With this paradigm, one of the proposed methods, RE-SVM, is based on constructing a SVM separating hyperplane, but simultaneously allowing observations to be relabeled during the training process. The results obtained by this method, in datasets in which noise was added to the training labels, showed that this strategy outperforms, in terms of accuracy, classical SVM and other SVM-based robust methodologies. See Bertsimas et al. (2019) for alternative robust classifiers under label noise.

In this chapter we propose a novel binary supervised classification method, called Optimal Classification Tree with Support Vector Machines (OCTSVM), that profits both from the ideas of SVM and OCT to build classification rules. Specifically, our method uses the hierarchical structure of OCT, which leads to easily interpretable rules, but splits are based on SVM hyperplanes, maximizing the margin between the two classes at each node of the tree. The fact that the combination of SVM and classification tree tools provides enhanced classifiers is not new. A similar approach can be found in Bennett and Blue (1998). Nevertheless, in that paper the authors analyze the greedy CART strategy by incorporating, sequentially the maximization of the margin, over known assignments of observations to the leaves of the tree. Opposite to that, OCTSVM does not assume those assumptions and it performs an exact optimization approach. Moreover, this new method also incorporates decisions on relabeling observations in the training dataset, making it specially suitable for datasets where adversary attacks are suspected. The results of our experiments show

that OCTSVM outperforms other existing methods under similar testing conditions. In contrast to the robust classifiers under label noise provided in Bertsimas et al. (2019), our method is not based on the worst-case paradigm commonly used in the field of robust optimization, but in the convenience of finding good classifiers under the presence of unknown noisy labels.

The rest of the chapter is organized as follows. In Section 5.2 we recall some of the notation used through the following sections. Section 5.3 is devoted to introduce our methodology, and presents a valid Mixed Integer Non Linear Programming formulation. In Section 5.4 we report the results obtained in our computational experiments, in particular, the comparison of our method with OCT, OCT-H and the greedy CART. Finally, some conclusions and further research on the topic are drawn in Section 5.5.

5.2 Preliminaries

In this section we recall the main elements in the approach that will be presented in Section 5.3 which allows us to construct robust classifiers under label noises, namely, RE-SVM and OCT-H.

On the one hand, we saw in previous chapter that given a training sample of a binary classification problem in the form $\mathcal{X} = \{(x_1, y_1),\ \ldots, (x_n, y_n),\} \subseteq \mathbb{R}^p \times \{-1, +1\}$, and a separating hyperplane $\mathcal{H} = \{z \in \mathbb{R}^p : \omega' z + \omega_0 = 0\}$, RE-SVM can be formulated as follows:

$$
\begin{aligned}
\min\ & \frac{1}{2}\|\omega\|_2^2 + c_1 \sum_{i=1}^{n} d_i + c_2 \sum_{i=1}^{n} \xi_i && && \text{(RE} - \text{SVM)}\\
\text{s.t.}\ & y_i(\omega' x_i + \omega_0) - 2y_i(\beta_i' x_i + \beta_{i0}) \geq 1 - d_i, && \forall i = 1, \ldots, n,\\
& \beta_{ij} = \xi_i \omega_j, && \forall i = 1, \ldots, n, j = 0, \ldots, p,\\
& \omega \in \mathbb{R}^p,\ \omega_0 \in \mathbb{R}, && \\
& \beta_i \in \mathbb{R}^p,\ \beta_{i0} \in \mathbb{R}, && \forall i = 1, \ldots, n,\\
& d_i \in \mathbb{R}_+,\ \xi_i \in \{0, 1\}, && \forall i = 1, \ldots, n.
\end{aligned}
$$

where ξ_i takes value 1 if the ith observation of the training sample is relabelled, and 0 otherwise and d_i is the misclassifying error defined as the hinge loss:

$$
d_i = \begin{cases} \max\{0, 1 - y_i(\omega' x_i + \omega_0)\} & \text{if observation } i \text{ is not relabelled} \\ \max\{0, 1 + y_i(\omega' x_i + \omega_0)\} & \text{if observation } i \text{ is relabelled} \end{cases},
$$

for $i = 1, \ldots, n$. The costs parameters c_1 and c_2 (unit cost per misclassifying error and per relabelled observation) allow one to find a trade-off between large separation

between classes: c_1 and c_2 are parameters modelling the unit cost of misclassified errors and relabelling, respectively. ($\|\cdot\|_2$ stands for the Euclidean norm in $\mathbb{R}^p$.)

On the other hand, Classification Trees (CT) are a family of classification methods based on a hierarchical relation among a set of nodes. The decision rule for CT methods is built by recursively partitioning the feature space by means of hyperplanes. At the first stage, a root node for the tree is considered where all the observations belongs to. Branches are sequentially created by splits on the feature space, creating intermediate nodes until a leaf node is reached. Then, the predicted label for an observation is given by the majority class of the leaf node where it belongs to.

Specifically, at each node, t, of the tree a hyperplane $\mathcal{H}_t = \{z \in \mathbb{R}^p : \omega_t' z + \omega_{t0} = 0\}$ is constructed and the splits are defined as $\omega_t' z + \omega_{t0} < 0$ (left branch) and $\omega_t' z + \omega_{t0} \geq 0$ (right branch). In Fig. 5.1 we show a simple classification tree with depth two, for a small dataset with 6 observations, that are correctly classified on the leaves.

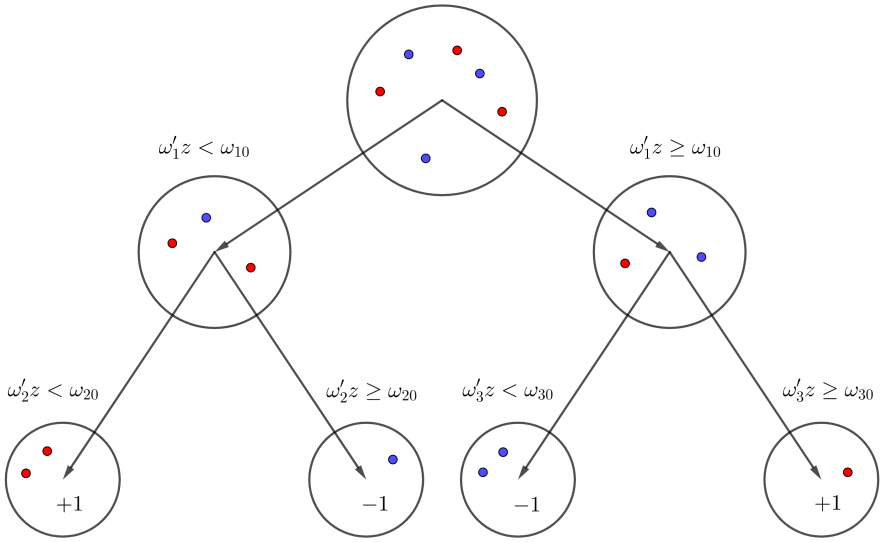


Figure 5.1: Decision tree of depth two.

The most popular method to construct Classification Trees from a training dataset is CART, introduced by Breiman et al. (1984). CART is a greedy heuristic approach, which myopically constructs the tree without further foreseen to deeper nodes. Starting at the root node, it decides the splits by means of hyperplanes minimizing an impurity function in each node. Each split results in two new nodes, and this procedure is repeated until a stopping criterion is reached (maximal depth, minimum number of observations in the same node, etc). Deep CART trees may

lead to overfitting in out-of-sample observations, and therefore trees are normally subject to a prune process based on the trade-off between the impurity function reduction and a cost-complexity parameter. The main advantage of CART is that it is easy to implement and fast to train.

On the other hand, Bertsimas and Dunn (2017) propose an optimal approach to build CT by solving a Mathematical Programming problem which builds the decision tree in a compact model considering its whole structure and at the same time making decisions on pruning or not pruning the branches.

Given a maximum depth, D, for the Classification Tree it can have at most $T = 2^{D+1} - 1$ nodes. These nodes are differentiated in two types:

- Branch nodes: $\tau_B = \{1, \ldots, \lfloor T/2 \rfloor\}$ are the nodes where the splits are applied.
- Leaf nodes: $\tau_L = \{\lceil T/2 \rceil, \ldots, T\}$ are the nodes where predictions for observations are performed.

We use the following notation concerning the hierarchical structure of a tree:

- $p(t)$: parent of node t, for $t = 1, \ldots, T$.
- τ_{bl}: set of nodes that follow the left branch on the path from their parent nodes. Analogously, we define τ_{br} as the set of nodes whose right branch has been followed on the path from their parent nodes.
- u: set of nodes that have the same depth inside the tree. We represent by U the whole set of levels. The root node is the zero-level, u_0, hence, for a given depth D we have $D + 1$ levels, being u_D the set of leaf nodes.

OCTs are constructed by minimizing the following objective function:

$$\sum_{t \in \tau_L} L_t + c \sum_{t \in \tau_B} \delta_t,$$

where L_t stands for the misclassification errors at the leaf t (measured as the number of wrongly classified observations in the leaf), and δ_t is a binary variable that indicates if a split is produced at t. Therefore, the constant c is used to regulate the trade-off between the complexity (depth) and the accuracy (misclassifying errors of the training sample) of the tree. In its simplest version, motivated by what it is done in CART, the splits are defined by means of a single variable, i.e., in the form $x_j \leq \omega_{j0}$. Nevertheless, OCT can be extended to a more complex version where the splits are hyperplanes defined by their normal vector, $a \in \mathbb{R}^p$ which is known as OCT-H. Moreover, a robust version of OCT has also been studied under the noise label scenario Bertsimas et al. (2019).

5.3 Optimal Classification Trees with SVM splits and Relabeling (OCTSVM)

This section is devoted to introduce our new classification methodology, namely OCTSVM. The rationale of this approach is to combine the advantage of hierarchical classification methods such as Classification Trees, with the benefits from using distance-based classification errors, by means of hyperplanes maximizing the margin between them (SVM paradigm). Therefore, this new model rests on the idea of constructing an optimal classification tree in which the splits of the nodes are performed by following the underlying ideas of model RE-SVM: 1) the splits are induced by hyperplanes in which the positive and the negative classes are separated maximizing the margin between classes, 2) minimizing the classification errors, and 3) allowing observations to be relabeled along the training process. In contrast to what it is done in other Classification Tree methods, OCTSVM does not make a distinction (beyond the hierarchical one) between branch and leaf nodes, in the sense that RE-SVM based splits are sequentially applied in each node, and the final classification for any observation comes from the hyperplanes resulting at the last level of the tree, in case there are no pruned branches, or at the last node where a split was made in case of a pruned branch.

As it has already been pointed out, OCT-H is a classification tree that allows the use of general (oblique) hyperplane splits, which is built by solving a single optimization problem that takes into account the whole structure of the tree. Nevertheless, despite the good results this method has proven to obtain in real classification problems, a further improvement is worth to be considered. In Figure 5.2 we see a set of points in the plane differentiated by geometrical elements (triangles and circles) in two classes. Looking at the left picture, one can see one of the optimal solutions of OCT-H for depth equal to two, where the red hyperplane is the split applied at the root node and the black ones are applied on the left and right descendants, which define the four leaves. This solution is optimal, for a certain value of the cost-complexity parameters, since it does not make any mistakes on the final classification. Nevertheless, since this method does not have any kind of control on the distances from points to the hyperplanes, one can observe that the blue class has very tiny margins at the leaves, and hence, for this class, misclassification errors are more likely to occur in out-of-sample observations. On the other hand, on the right side of Figure 5.2 one sees another possible optimal solution for the OCTSVM model with depth equal to one (note that unlike OCT-H, OCTSVM constructs a final SVM-based classifier at each of the leaf nodes, which may be identified with an extra depth). Again, the red hyperplane is the split applied at the root node and the black ones are the classification splits applied at the two leaves. Despite these

two methods are obtaining a perfect classification on the training sample, Figure 5.2 shows that OCTSVM provides a more balanced solution than OCT-H since it has wider margins between both classes, what could be translated into a higher accuracy for out-of-sample observations.

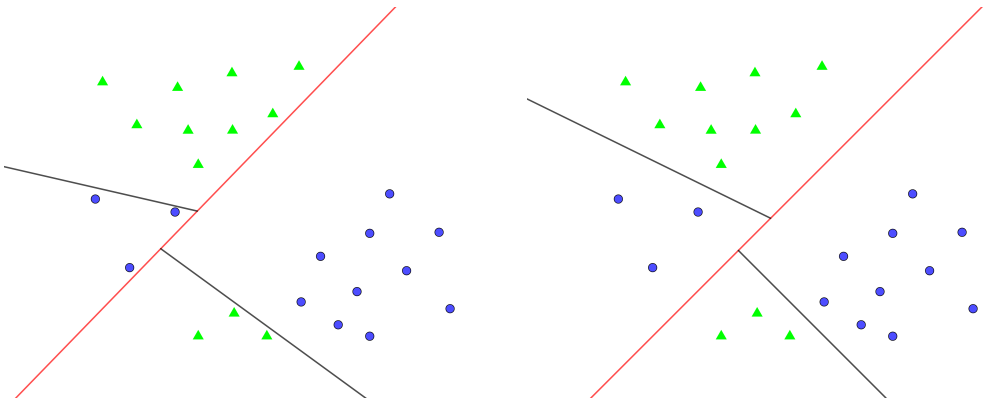


Figure 5.2: Optimal solutions for OCT-H with $D = 2$ (left) and OCTSVM with $D = 1$ (right).

In order to formulate the OCTSVM as a MINLP, we will start describing the rationale of its objective function that must account for the margins induced by the splits at all nodes, the classification errors, the penalty paid for relabelling observations and the cost-complexity of the final classification tree. To formulate the above concepts we need different sets of variables. First of all, we consider continuous variables: $\omega_t \in \mathbb{R}^p,\ \omega_{t_0} \in \mathbb{R},\ t = 1, \ldots, T$, which represent the coefficients and the intercept of the hyperplane performing the split at node t. Taking into account that the margin of the hyperplane $\mathcal{H}_t = \{z : \omega_t' z + \omega_{t0} = 0\}$ is given by $\frac{2}{||\omega_t||}$, maximizing the minimum margin between classes induced by the splits can be done introducing an auxiliary variable $\iota \in \mathbb{R}$ (that will be minimized in the objective function) which is enforced by the following constraints:

$$\frac{1}{2}||\omega_t||_2 \leq \iota \qquad \forall t = 1, \ldots, T. \tag{5.1}$$

Once the maximization of the margin is set, we have to model the minimization of the errors at the nodes, whereas at the same time we minimize the number of relabelled observations. These two tasks are accomplished by the variables $d_{it} \in \mathbb{R},\ i = 1, \ldots, n,\ t = 1, \ldots, T$, that account for the misclassification error of observation i at node t, and $\xi_{it} \in \{0, 1\}$ binary variables modelling whether observation i is relabeled or not at node t. If c_1 and c_2 are the unit costs of misclassification and relabelling, respectively, our goal is achieved adding to the objective function

the following two terms:

$$c_1 \sum_{i=1}^{n} \sum_{t=1}^{T} d_{it} + c_2 \sum_{i=1}^{n} \sum_{t=1}^{T} \xi_{it}.$$

The correct meaning of these two sets of variables must be enforced by some families of constraints that we describe next. Nevertheless, for the sake of readability before describing those constraints modeling these d_{it} and ξ_{it}, we must introduce another family of variables the $\beta_{it} \in \mathbb{R}^p, \beta_{i0} \in \mathbb{R},\ i = 1, \ldots, n,\ t = 1, \ldots, T$, which are continuous variables equal to the coefficients of the separating hyperplane at node t when observation i is relabelled, and equal to zero otherwise. In addition, we consider binary variables $z_{it} \in \{0, 1\}$ needed to control whether observation i belongs to node t of the tree. Now, putting all these elements together, as it is done in RE-SVM, we can properly define the splits and their errors at each node of the tree using the following constraints:

$$y_i(\omega_t' x_i + \omega_{t0}) - 2y_i(\beta_t' x_i + \beta_{it0}) \geq 1 - d_{it} - M(1 - z_{it}), \quad \begin{cases} \forall i = 1, \ldots, n, \\ t = 1, \ldots, T, \end{cases} \tag{5.2}$$

$$\beta_{itj} = \xi_{it}\omega_{tj}, \quad \forall i = 1, \ldots, n,\ t = 1, \ldots, T,\ j = 0, \ldots, p. \tag{5.3}$$

Constraints (5.3), which can be easily linearized, are used to define the β_{it} variables: they equal the ω_t variables when the observation i is relabelled ($\xi_i = 1$), and are equal to zero otherwise ($\xi_i = 0$). On the other hand, constraints (5.2) control the relabelling at each node of the tree. If an observation i is in node t ($z_{it} = 1$), and $\xi_i = 0$, we obtain the standard SVM constraints for the separating hyperplane. Nevertheless, if $\xi_i = 1$, then the separating constraints are applied for the observation i as if its class were the opposite to its actual one, i.e., as if observation i is relabeled. Moreover, since M is a big enough constant, these constraints do not have any kind of impact in the error variables of node t if observation i does not belong to this node ($z_{it} = 0$).

On the other hand, there are still some details left that must be imposed to make the model to work as required. In the decision tree, observations start at the root node and they advance descending through the levels of the tree until they reach a leaf or a pruned node. Hence, we have to guarantee that observations must belong to one, and only one, node per level. By means of the z_{it} variables, this can be easily done by the usual assignment constraints applied in each level, $u \in U$, of the tree:

$$\sum_{t \in u} z_{it} = 1, \qquad \forall i = 1, \ldots, n,\ u \in U. \tag{5.4}$$

Moreover, for consistency in the relation between a node and its ancestor, it is clear that if observation i is in node t ($z_{it} = 1$), then, observation i must be also in the parent of node t ($z_{ip(t)} = 1$), with the only exception of the root node. Besides, if observation i is not in node t ($z_{it} = 0$), then i can not be in its successors, and this is modeled by adding the following constraints to the problem:

$$z_{it} \leq z_{ip(t)}, \qquad \forall i = 1, \ldots, n,\ t = 2, \ldots, T. \tag{5.5}$$

So far, the OCTSVM model has everything it needs to properly perform the splits by following the RE-SVM rationale described in the previous chapter, taking into consideration the tree complexity, and maintaining the hierarchical relationship amongst nodes. The last element that we need to take care of, to assure the correct performance of the whole model, is to define how observations follow their paths inside the tree. We get from constraints (5.5) that observations move from parent to children (nodes), but every non terminal node has a left and a right child node, and we need to establish how observations take the left or the right branch. Since the splits are made by the separating hyperplane, we force observations that lie on the positive half space of a hyperplane to follow the right branch of the parent node, and observations that lie on the negative one to take the left branch. This behavior is modeled with the binary variables $\theta_{it} \in \{0, 1\}$, that are used to identify whether observation i lies in the positive half space of the separating hyperplane at node t, $\theta_{it} = 1$, or if observation i lies on the negative half space, $\theta_{it} = 0$. By considering M a big enough constant, the correct behavior of the path followed by the observations is enforced by the following constraints:

$$\omega_t' x_i + \omega_{t0} \geq -M(1 - \theta_{it}), \qquad \forall i = 1, \ldots, n,\ t = 1, \ldots, T, \tag{5.6}$$

$$\omega_t' x_i + \omega_{t0} \leq M\theta_{it}, \qquad \forall i = 1, \ldots, n,\ t = 1, \ldots, T. \tag{5.7}$$

Hence, by making use of these θ_{it} variables, and distinguishing between nodes that come from left splits, τ_{bl} (nodes indexed by even numbers), and right splits, τ_{br} (nodes indexed by odd numbers), we control that the observations follow the paths through the branches in the way we described above throughout the following constraints:

$$z_{ip(t)} - z_{it} \leq \theta_{ip(t)}, \qquad \forall i = 1, \ldots, n,\ t \in \tau_{bl}, \tag{5.8}$$

$$z_{ip(t)} - z_{it} \leq 1 - \theta_{ip(t)}, \qquad \forall i = 1, \ldots, n,\ t \in \tau_{br}. \tag{5.9}$$

According to constraints (5.8), if an observation i is on the parent node of an even node t ($z_{ip(t)} = 1$), and i lies on the negative half space of the hyperplane defining the split on $p(t)$ ($\theta_{ip(t)} = 0$), then z_{it} is forced to be 1. Hence, $\theta_{ip(t)} = 0$ implies that

observation i takes the left branch to the child node $t \in \tau_{bl}$. Moreover, we can see that this constraint is consistent since if $z_{ip(t)} = 1$, but observation i is not in the left child node, $z_{it} = 0$, $t \in \tau_{bl}$, then $\theta_{ip(t)}$ equals 1, what means that observation i lies on the positive half space of the hyperplane of $p(t)$. On the other hand, constraints (5.9) are similar but for the right child nodes, τ_{br}. If an observation i is in the parent node of an odd node $t \in \tau_{br}$, and i lies on the positive half space of the hyperplane of $p(t)$ ($\theta_{ip(t)} = 1$), then, $z_{it} = 1$ what means that observation i has to be on node t.

The final term to be included in the objective function of the problem is the complexity of the resulting tree. Following the approach in OCT and OCT-H, we consider binary variables $\delta_t \in \{0, 1\}$, $t = 1, \ldots, T$, that control whether a separating split is applied at node t. Thus, to control the tree complexity resulting of the process, we minimize the sum of these variables multiplied by a cost penalty c_3. Gathering all the components together, the objective function to be minimized in our problem results in

$$\iota + c_1 \sum_{i=1}^{n} \sum_{t=1}^{T} d_{it} + c_2 \sum_{i=1}^{n} \sum_{t=1}^{T} \xi_{it} + c_3 \sum_{t=1}^{T} \delta_t.$$

According to this, it is important to make a distinction between the methods that use, or the ones that do not use, SVM based splits. When a model does not use SVM based splits, taking into account that the binary variable $\delta_t = 1$ implies that a hyperplane is being used to split the points in node t, complexity can be easily regulated by just imposing $\|\omega_t\|_2 \leq M\delta_t$ in all the branch nodes. Nevertheless, when using a SVM based method such as the one we are presenting here, the previous constraint would be in conflict with constraints (5.2). This is due to the fact that $\delta_t = 0$ would imply $\|\omega_t\|_2 = 0$, and under this scenario, observations in this node would have to pay for the margin violation error $d_{it} = 1$, even though these errors are not justified since points are not being separated at this node. To overcome this issue, we need to add some other constraints to the model, but before getting into the mathematical formulation, we would like to emphasize the different aspects that explain the difficulty of the problem. The objective function accounts for the complexity cost of node t ($\delta_t = 1$) when a hyperplane is actually built so as to split the points in node t. In case we do not pay the complexity cost in node t ($\delta_t = 0$), it does not necessarily mean that a hyperplane is not built at this node, it could simply turn out to be a hyperplane that leaves all the observations at one of the half spaces it creates, i.e., a hyperplane that is not splitting the points. These non splitting hyperplanes do not affect the training set at all, and more importantly, they would not affect out of sample observations since predictions will occur in leaf nodes or in the first branch node in which $\delta_t = 0$. Going back to the formulation, the first

step now is to introduce some binary variables, $h_{it} \in \{0,1\},\ i = 1,\dots,n,\ t \in \tau_b$, that will be relaxed afterwards, defined by $h_{it} = z_{it}\theta_{it}$. These variables tell us which observations belong to node t and at the same time lie on the positive half space of the split created in t. It is important to define these variables because they are used to measure whether the hyperplane is splitting the points in node t or not. This is going to be done by means of some new binary variables $v_t \in \{0,1\},\ t \in \tau_b$, that will be equal to zero in case all the points in node t belong to the positive half space of the hyperplane built in this node, $\mathcal{H}_t$, and one otherwise. For the definition of the δ_t variables to make sense, the following constraints must simultaneously be considered:

$$h_{it} = z_{it}\theta_{it}, \qquad \forall i = 1,\dots,n,\ t = 1,\dots,T, \tag{5.10}$$

$$\sum_{i=1}^{n}(z_{it} - h_{it}) \leq M v_t, \qquad \forall t = 1,\dots,T, \tag{5.11}$$

$$\sum_{i=1}^{n} h_{it} - M(1 - v_t) \leq M\delta_t, \qquad \forall t = 1,\dots,T, \tag{5.12}$$

where M is a big enough constant. The reader should observe that (5.10) is the definition of the h_{it} variables as the product of z_{it} by θ_{it}. These constraints can be easily linearized by the usual tricks. On the other hand, taking into account that v_t variables would tend to be zero (by the effects on constraints (5.12)), constraints (5.11) stand for the definition of the v_t variables, since these variables could be equal to zero just in case all the h_{it} are equal to one, what means that all the observations belong to the positive half space of $\mathcal{H}_t$. Finally, we obtain through constraints (5.12) the definition of the δ_t variables. Since these variables are being minimized in the objective function, they would try to be equal to zero. Nevertheless, this can just happen if $v_t = 0$, what means that all the observations belong to the positive half space of $\mathcal{H}_t$, or if $\sum_{i=1}^{n} h_{it} = 0$ in case $v_t = 1$, what means that all the observations belong to the negative half space of $\mathcal{H}_t$. Hence, δ_t is equal to one if and only if the points in node t are being actually separated by $\mathcal{H}_t$.
Finally, another point that it is important to remark about the δ_t variables is that once a non-leaf node does not split (that is, the corresponding branch is pruned at this node), the successors of node t can not make splits either to maintain the correct hierarchical structure of the tree. Recalling that $p(t)$ is the parent node of node t, we can guarantee this relationship throughout the following constraints

$$\delta_t \leq \delta_{p(t)}, \qquad \forall t = 2,\dots,T. \tag{5.13}$$

Some of the constraints discussed above require big-M constants. The value of

these constants can be adjusted once we know the dimension (n, p) of the dataset involved in the optimization problem. Recall that $x_i \in [0,1]^p$, therefore it is well known that the maximum distance between two points in such a domain is $\sqrt{p}$. Hence, the maximum distance from a point to a hyperplane is also $\sqrt{p}$. According to this, the big-M constant in constraints (5.2), (5.5) and (5.6) is actually $\sqrt{p}$.
On the other hand, the h_{it} variables are upper bounded by one, and therefore $\sum_{i=1}^{n} h_{it}$ is upper bounded by n. As a result, the big-M constant involved in constraints (5.10) and (5.11) is n as well.
Gathering all the constraints together, and substituting the generic big-M constants by those estimated in our discussion above, the OCTSVM is obtained by solving the following MINLP:

$$\min\ \iota + c_1 \sum_{i=1}^{n}\sum_{t=1}^{T} d_{it} + c_2 \sum_{i=1}^{n}\sum_{t=1}^{T} \xi_{it} + c_3 \sum_{t=1}^{T} \delta_t \qquad \text{(OCTSVM)}$$

$$\text{s.t. } \frac{1}{2}||\omega_t||_2 \leq \iota, \quad \forall t = 1,\dots,T,$$

$$y_i(\omega_t' x_i + \omega_{t0}) - 2y_i(\beta_t' x_i + \beta_{it0}) \geq 1 - d_{it} - \sqrt{p}(1 - z_{it}), \qquad \forall i = 1,\dots,n,\ t = 1,\dots,T,$$

$$\beta_{itj} = \xi_{it}\omega_{tj}, \qquad \forall i = 1,\dots,n,\ t = 1,\dots,T,\ j = 0,\dots,p, \tag{5.14}$$

$$\sum_{t\in u} z_{it} = 1, \qquad \forall i = 1,\dots,n,\ u \in U,$$

$$z_{it} \leq z_{ip(t)}, \qquad \forall i = 1,\dots,n,\ t = 2,\dots,T,$$

$$\omega_t' x_i + \omega_{t0} \geq -\sqrt{p}(1 - \theta_{it}), \qquad \forall i = 1,\dots,n,\ t = 1,\dots,T,$$

$$\omega_t' x_i + \omega_{t0} \leq \sqrt{p}\theta_{it}, \qquad \forall i = 1,\dots,n,\ t = 1,\dots,T,$$

$$z_{ip(t)} - z_{it} \leq \theta_{ip(t)}, \qquad \forall i = 1,\dots,n,\ t \in \tau_{bl},$$

$$z_{ip(t)} - z_{it} \leq 1 - \theta_{ip(t)}, \qquad \forall i = 1,\dots,n,\ t \in \tau_{br},$$

$$h_{it} = z_{it}\theta_{it}, \qquad \forall i = 1,\dots,n,\ t = 1,\dots,T, \tag{5.15}$$

$$\sum_{i=1}^{n}(z_{it} - h_{it}) \leq n v_t, \qquad \forall t = 1,\dots,T,$$

$$\sum_{i=1}^{n} h_{it} - n(1 - v_t) \leq n\delta_t, \qquad \forall t = 1,\dots,T,$$

$$\delta_t \leq \delta_{p(t)}, \qquad \forall t = 2,\dots,T,$$

$$d_{it} \in \mathbb{R}^+, \beta_{it} \in \mathbb{R}^p, \beta_{it0} \in \mathbb{R}, \xi_{it}, z_{it}, \theta_{it}, h_{it} \in \{0,1\}, \forall i = 1,\dots,n,\ t = 1,\dots,T,$$

$$\omega_t \in \mathbb{R}^p, \omega_{t0} \in \mathbb{R}, \delta_t, v_t \in \{0,1\}, \forall t = 1,\dots,T,$$

$$\iota \in \mathbb{R}.$$

The reader may note that the above formulation has two families of bilinear

constraints, namely (5.14) and (5.15), that can be easily linearized giving rise to a mixed integer second order cone formulation.

5.4 Experiments

In this section we present the results of our computational experiments. Five different classification tree-based methods are compared, CART, OCT, OCT-H, OCT+SVM and OCTSVM, on nine popular real-life datasets from UCI Machine Learning Repository Lichman et al. (2013). Notice that OCT+SVM is a modification of our OCTSVM in which the ξ_{it} variables are fixed to zero, i.e., no relabeling is allowed in the model. This method is included so as to assess the isolated effect of using margin-based splits, without relabeling, within the classification trees. The considered datasets together with their names and dimensions (n: number of observations, p: number of features) are reported in the three tables of this section.

Our computational experiments focus on the analysis of the accuracy and AUC (area under the ROC curve) of the different classification tree-based methods. This analysis is based in four different experiments for each one of the nine considered dataset. Our goal is to analyze the usefulness of the different methods for classifying data affected by label noise. Therefore, in our experiments we use, apart from the original datasets, three different modifications where in each one of them a percentage of the labels in the sample are randomly flipped. The percentages of flipped labels range in $\{0\%, 20\%, 30\%, 40\%\}$, where 0% indicates that the original training data set is used to construct the tree.

We perform a 4-fold cross-validation scheme, i.e., datasets are split into four random train-test partitions. One of the folds is used for training the model while the rest are used for testing. When testing the classification rules, we compute the accuracy, in percentage, on out of sample data, as well as the AUC, which provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

The CART method was coded in `R` using the `rpart` library. OCT, OCT-H, OCT+SVM and OCTSVM Mathematical Programming models were coded in `Python` and solved using `Gurobi` 8.1.1 on a PC Intel Core i7-7700 processor at 2.81 GHz and 16GB of RAM. A time limit of 30 seconds was set for training. Not all the problems were solved to optimality, nevertheless, this time limit was enough in order to obtain good classifiers. The average gap of the experiments is reported in table 5.3.

The calibration of the parameters of the different optimization-based models compared in these experiments was set as follows:

- For OCTSVM we used a grid on $\left\{10^i : i = -5, \ldots, 5\right\}$ for the constants c_1 and c_2, and a smaller one $\left\{10^i : i = -2, \ldots, 2\right\}$ for c_3. For OCT+SVM c_2 is not used since all the ξ_{it} variables are set to zero.

- For OCT, in order to check every possible optimal subtree of a maximal tree for a given depth D, we did the predictions in two steps. We first set up a grid on the parameter $c_1 = \left\{1, \ldots, 2^D - 1\right\}$ and solved a slightly different OCT model in which the objective function did not take into account the complexity term, since complexity was already considered in the model by adding the following constraint $\sum_{t \in \tau_B} \delta_t \leq c_1$. The resulting solutions that were optimal for a certain c_1 of the original OCT problem were afterwards computed. This methodology was not used in OCT-H since the grid of the modified problem should have been extended to $c_1 = \left\{1, \ldots, p(2^D - 1)\right\}$, therefore we used the same grid that we used for c_1 in OCTSVM directly into OCT-H formulation. On the other hand, the minimum number of observations per leaf was set to a 5% of the training sample size.

- CART trees were post-processed to satisfy any depth constraint as it was done with OCT. The minimum number of observations per leaf was the same 5% of the training sample size that we used for OCT and OCT-H.

Last to mention, the depth, D, considered in these experiments was set equal to three for CART, OCT and OCT-H, whereas for OCTSVM and OCT+SVM we fixed depth equal to two, creating consequently trees with 3 levels, to set a fair comparison amongst the different methods.

For each dataset, we replicate each experiment four times. In Table 5.1 we show the average (± standard deviation) accuracies for each one of the methods and datasets. In addition, Table 5.2 shows the average (± standard deviation) AUC results. The best results are highlighted (boldfaced). The first column stands for the percentage of flipped labels (FL) of the training sets. On the other hand, the last column shows the mean difference (Diff) between OCTSVM and the best result amongst OCT and OCT-H. Finally, in Table 5.3 we report the average MINLP gaps obtained by the models at the end of the training time limit.

5.5 Conclusions

One of the classification methods that has experienced a more in depth transformation in the last years is classification trees. Since the pioneer contribution by Breiman et al. Breiman et al. (1984), where CART was proposed, this technique has included tools from mathematical optimization giving rise to the so called OCT

	% FL	CART	OCT	OCT-H	OCT+SVM	OCTSVM	Diff
Australian (690,14)	0	84.91 ± 1.31	85.48 ± 0.99	85.16 ± 1.28	85.62 ± 1.08	**86.16 ± 0.68**	0.68 ± 0.76
	20	83.35 ± 5.30	**85.31 ± 1.12**	80.05 ± 4.57	81.93 ± 7.83	85.25 ± 1.34	-0.06 ± 1.03
	30	70.50 ± 11.89	79.27 ± 7.55	72.01 ± 4.25	75.24 ± 10.46	**82.57 ± 7.46**	3.29 ± 11.96
	40	54.34 ± 7.94	70.36 ± 9.56	64.63 ± 4.25	64.38 ± 7.93	**77.92 ± 8.19**	7.56 ± 11.71
Banknote (1372,5)	0	91.64 ± 2.09	87.94 ± 1.54	**98.60 ± 0.47**	92.68 ± 4.86	96.06 ± 3.19	-2.54 ± 3.32
	20	88.00 ± 2.63	85.99 ± 1.91	70.01 ± 9.18	66.46 ± 10.01	**89.97 ± 4.71**	3.97 ± 4.63
	30	85.79 ± 3.38	**85.96 ± 2.27**	63.18 ± 9.91	67.36 ± 12.85	84.03 ± 12.46	-1.93 ± 12.72
	40	69.13 ± 10.33	77.73 ± 9.25	59.15 ± 8.53	58.45 ± 12.66	**87.16 ± 12.46**	9.42 ± 8.67
BreastCancer (683,9)	0	91.81 ± 1.51	93.71 ± 0.94	94.21 ± 1.06	**96.15 ± 0.94**	95.09 ± 5.27	2.44 ± 1.07
	20	90.38 ± 2.63	**92.88 ± 1.72**	89.45 ± 4.24	82.96 ± 5.29	91.85 ± 3.07	-1.03 ± 3.08
	30	87.52 ± 3.38	**92.15 ± 1.84**	84.41 ± 5.41	79.16 ± 4.94	89.90 ± 2.75	-2.24 ± 3.85
	40	73.95 ± 12.53	**90.44 ± 2.68**	75.36 ± 6.81	74.58 ± 5.15	87.28 ± 4.71	-3.15 ± 5.26
Heart (270,13)	0	72.34 ± 3.57	75.02 ± 1.82	78.95 ± 2.29	82.31 ± 2.05	**83.36 ± 1.41**	4.41 ± 1.88
	20	70.53 ± 6.16	74.10 ± 3.57	74.66 ± 4.24	73.95 ± 5.98	**80.13 ± 4.39**	5.46 ± 4.92
	30	71.87 ± 3.72	71.87 ± 6.23	71.19 ± 5.61	73.7 ± 4.69	**77.25 ± 6.31**	5.37 ± 7.24
	40	71.72 ± 3.63	64.18 ± 5.32	65.18 ± 5.04	66.82 ± 5.15	**74.26 ± 3.39**	9.07 ± 4.94
Ionosphere (351,34)	0	81.09 ± 4.73	85.34 ± 2.36	84.77 ± 1.97	83.15 ± 4.46	**85.55 ± 2.65**	0.20 ± 2.70
	20	74.46 ± 6.77	**78.74 ± 3.14**	74.88 ± 5.78	77.06 ± 3.63	78.05 ± 4.65	-0.68 ± 5.43
	30	65.80 ± 6.76	74.30 ± 5.70	73.61 ± 4.47	74.84 ± 4.46	**76.83 ± 5.21**	2.53 ± 7.27
	40	60.48 ± 7.39	70.89 ± 5.30	66.06 ± 6.69	71.73 ± 4.64	**75.68 ± 2.62**	4.79 ± 5.77
MONK's (415,17)	0	59.19 ± 3.04	60.30 ± 2.90	61.24 ± 2.40	60.48 ± 2.59	**62.85 ± 1.72**	1.61 ± 2.90
	20	55.40 ± 7.01	59.59 ± 3.42	58.96 ± 4.93	59.49 ± 1.77	**62.40 ± 2.15**	2.80 ± 3.79
	30	55.10 ± 4.88	57.90 ± 5.00	59.21 ± 2.91	58.73 ± 0.97	**60.57 ± 4.67**	1.36 ± 4.93
	40	50.30 ± 5.99	56.86 ± 3.04	56.86 ± 3.61	58.84 ± 2.11	**61.91 ± 2.30**	5.05 ± 4.41
Parkinson (240,40)	0	69.19 ± 8.95	76.09 ± 4.82	74.82 ± 3.18	80.66 ± 2.26	**81.67 ± 4.26**	5.57 ± 4.26
	20	61.95 ± 12.39	73.95 ± 4.34	66.40 ± 5.20	74.96 ± 4.69	**77.29 ± 3.88**	3.33 ± 5.27
	30	57.82 ± 10.61	71.05 ± 7.03	62.63 ± 6.46	70.27 ± 5.42	**75.57 ± 3.70**	4.52 ± 6.56
	40	51.76 ± 8.18	67.95 ± 7.44	59.39 ± 3.98	62.38 ± 5.51	**69.59 ± 5.10**	1.64 ± 9.46
Sonar (208,60)	0	62.11 ± 10.86	69.17 ± 4.82	71.53 ± 4.69	74.15 ± 3.70	**76.86 ± 3.86**	5.32 ± 6.16
	20	58.06 ± 9.03	64.41 ± 5.65	67.41 ± 4.07	75.21 ± 5.59	**71.62 ± 4.29**	4.20 ± 4.84
	30	53.35 ± 6.40	60.62 ± 4.78	65.11 ± 5.76	67.36 ± 7.27	**70.07 ± 4.36**	4.96 ± 5.38
	40	50.04 ± 6.76	58.79 ± 5.18	60.24 ± 4.15	62.44 ± 4.13	**64.95 ± 11.88**	4.71 ± 10.68
Wholesale (440,7)	0	89.61 ± 2.22	90.35 ± 1.28	**90.52** ± 1.27	87.54 ± 3.80	88.59 ± 1.82	-1.93 ±2.63
	20	81.22 ± 10.12	**87.80 ± 3.56**	85.14 ± 2.95	75.21 ± 5.59	81.63 ± 4.67	-6.17 ± 5.67
	30	77.40 ± 12.03	**83.99 ± 5.15**	78.01 ± 7.01	74.89 ± 6.35	78.95 ± 5.06	-5.03 ± 8.30
	40	61.46 ± 14.41	73.14 ± 16.11	72.31 ± 5.70	72.01 ± 10.52	**76.06 ± 4.94**	3.76 ± 16.90

Table 5.1: Average accuracies (± standard deviations) obtained in our computational experiments.

	% FL	CART	OCT	OCT-H	OCT+SVM	OCTSVM	Diff
Australian (690,14)	0	85.31 ± 1.72	86.10 ± 0.93	85.18 ± 1.52	85.92 ± 1.16	**86.30 ± 0.78**	0.20 ± 0.70
	20	83.78 ± 5.81	**85.88 ± 1.23**	80.33 ± 5.19	81.87 ± 9.42	85.52 ± 8.77	-0.35 ± 1.15
	30	70.26 ± 12.49	79.44 ± 7.70	71.87 ± 4.32	74.89 ±11.96	**82.70 ± 8.77**	3.26 ± 13.08
	40	53.98 ± 6.12	70.04 ± 10.26	64.91 ± 5.58	63.07 ± 9.80	**77.39 ± 9.39**	7.35 ± 12.53
Banknote (1372,5)	0	91.37 ± 2.15	87.52 ± 1.47	**98.56 ± 0.43**	90.84 ± 7.11	96.01 ± 3.23	-2.55 ± 3.34
	20	87.80 ± 2.69	85.87 ± 1.76	66.91 ± 10.95	63.38 ± 12.15	**89.47 ± 5.26**	3.60 ± 5.17
	30	85.33 ± 3.90	**85.54 ± 2.40**	60.13 ± 11.04	64.97 ± 13.83	84.17 ± 11.63	-1.37 ± 11.84
	40	67.54 ± 12.06	76.64 ± 10.95	56.05 ± 8.14	55.68 ± 12.33	**85.85 ± 10.02**	9.20 ± 9.98
BreastCancer (683,9)	0	91.36 ± 1.92	93.10 ± 1.33	93.60 ± 1.54	**95.73 ± 1.27**	94.20 ± 7.98	2.62 ± 1.41
	20	89.62 ± 2.86	**91.95 ± 2.46**	88.52 ± 5.76	79.58 ± 8.69	90.32 ± 3.40	-1.09 ± 4.14
	30	85.28 ± 8.18	**90.82 ± 2.46**	82.96 ± 8.50	74.00 ± 7.94	87.06 ± 4.37	-3.75 ± 6.23
	40	67.61 ± 15.04	**88.90 ± 3.96**	69.71 ± 10.28	67.46 ± 8.33	85.48 ± 7.57	-3.42 ± 9.10
Heart (270,13)	0	71.91 ± 2.89	74.49 ± 1.97	78.62 ± 2.19	82.17 ± 1.53	**82.87 ± 1.21**	4.24 ± 1.96
	20	69.88 ± 6.09	73.31 ± 3.42	74.24 ± 3.91	73.04 ± 6.08	**79.40 ± 4.29**	5.15 ± 4.43
	30	71.47 ± 3.04	71.16 ± 6.40	70.14 ± 5.82	72.75 ± 5.15	**73.97 ± 6.87**	4.73 ± 7.89
	40	71.45 ± 3.01	63.91 ± 5.34	64.07 ± 6.14	66.29 ± 5.85	**73.97 ± 3.08**	9.89 ± 6.29
Ionosphere (351,34)	0	78.29 ± 6.21	81.90 ± 3.84	81.46 ± 2.04	77.49 ± 5.66	**82.38 ± 3.71**	0.48 ± 4.26
	20	69.22 ± 9.11	**74.39 ± 3.87**	70.69 ± 6.64	73.18 ± 4.46	72.64 ± 7.39	-1.70 ±7.48
	30	60.11 ± 8.73	70.20 ± 6.56	69.09 ± 4.62	70.62 ± 6.42	**71.37 ± 5.07**	1.16 ± 7.39
	40	56.09 ± 7.09	68.23 ± 5.56	60.00 ± 7.84	64.32 ± 6.35	**69.44 ± 3.51**	1.21 ± 6.80
MONK's (415,17)	0	58.49 ± 4.06	59.83 ± 3.21	59.84 ± 3.85	59.90 ± 2.78	**60.92 ± 2.46**	1.08 ± 3.16
	20	54.93 ± 6.45	59.06 ± 3.96	58.29 ± 3.85	57.51± 2.83	**60.68 ± 2.36**	1.62 ± 4.50
	30	53.56 ± 5.26	57.25 ± 5.98	57.85 ± 2.95	55.86 ± 2.91	**59.58 ± 3.09**	1.73 ± 3.39
	40	49.51 ± 4.56	56.08 ± 3.41	55.67 ± 4.11	57.33 ± 2.57	**59.79 ± 2.96**	4.12 ± 3.81
Parkinson (240,40)	0	69.58 ± 8.13	76.15 ± 1.77	74.95 ± 3.05	80.77 ± 2.19	**81.81 ± 4.07**	5.66 ± 4.14
	20	62.69 ± 11.63	74.08 ± 4.34	66.48 ± 5.27	75.13 ± 4.72	**77.37 ± 3.47**	3.29 ± 5.34
	30	58.15 ± 10.43	70.89 ± 7.57	62.57 ± 6.71	70.18 ± 5.54	**75.87 ± 3.47**	4.97 ± 7.28
	40	51.94 ± 8.09	67.89 ± 7.61	59.36 ± 3.90	62.27 ± 5.98	**69.32 ± 5.30**	1.42 ± 9.52
Sonar (208,60)	0	62.11 ± 10.86	68.68 ± 4.65	71.19 ± 4.77	73.64 ± 4.25	**76.55 ± 3.68**	5.35 ± 6.34
	20	58.06 ± 9.03	63.75 ± 6.20	67.46 ± 5.67	68.81 ± 4.86	**71.61 ± 4.41**	4.15 ± 4.85
	30	53.35 ± 6.40	59.88 ± 5.54	65.20 ± 5.67	66.65 ± 7.78	**69.64 ± 4.42**	4.43 ± 5.10
	40	50.04 ± 6.76	58.00 ± 5.25	59.82 ± 4.98	62.05 ± 4.40	**64.99 ± 11.18**	5.17 ± 10.17
Wholesale (440,7)	0	89.61 ± 2.22	88.82 ± 2.47	**88.94 ± 3.02**	86.25 ± 3.58	87.36 ± 2.98	-1.57 ± 4.25
	20	81.22 ± 10.12	**85.80 ± 5.37**	82.41 ± 5.53	67.10 ± 12.64	77.12 ± 7.94	-8.67 ± 7.50
	30	77.40 ± 12.03	**81.80 ± 8.12**	74.57 ± 10.13	67.99 ± 12.29	72.40 ± 9.12	-9.39 ± 13.03
	40	61.46 ± 14.41	**72.30 ± 15.57**	64.72 ± 12.85	59.88 ± 10.52	69.66 ± 10.35	-2.64 ± 9.89

Table 5.2: Average AUC (± standard deviations) obtained in our computational experiments.

	% FL	OCT	OCT-H	OCT+SVM	OCTSVM
Australian (690,14)	0	12.50	85.36	27.33	91.59
	20	18.75	97.50	30.01	86.80
	30	68.75	99.23	45.31	82.92
	40	62.50	99.66	35.84	85.53
Banknote (1372,5)	0	100.00	78.00	76.64	82.53
	20	68.75	99.99	75.16	93.65
	30	56.25	99.99	76.69	91.75
	40	43.75	99.70	73.77	85.20
BreastCancer (683,9)	0	56.25	80.62	19.19	79.40
	20	56.04	87.50	54.07	78.17
	30	81.09	94.81	37.54	81.35
	40	82.89	98.26	23.57	86.62
Heart (270,13)	0	66.93	88.86	29.55	68.34
	20	54.68	93.78	45.09	81.01
	30	54.97	93.22	36.63	90.98
	40	99.06	96.82	32.54	85.05
Ionosphere (351,34)	0	68.33	75.00	47.23	84.15
	20	61.42	85.94	52.38	100.00
	30	80.80	89.63	74.39	84.16
	40	75.00	92.42	46.26	78.52
MONK's (415,17)	0	12.25	97.09	24.99	78.19
	20	37.22	97.03	18.68	68.73
	30	24.77	99.75	12.09	90.95
	40	62.01	96.41	21.62	83.78
Parkinson (240,40)	0	34.52	76.15	36.61	89.07
	20	55.92	79.91	40.15	85.52
	30	43.75	87.07	40.79	97.06
	40	36.97	88.49	41.49	84.54
Sonar (208,60)	0	50.00	75.63	19.88	77.73
	20	68.75	81.94	39.45	79.17
	30	68.75	83.00	30.48	88.75
	40	75.00	78.62	38.71	91.91
Wholesale (440,7)	0	31.25	38.67	28.71	32.34
	20	30.88	96.20	14.22	94.26
	30	31.25	95.07	7.39	91.25
	40	75.00	88.35	6.34	97.52

Table 5.3: Average MINLP Gaps of the optimization-based models to construct classification trees, within the time limit.

Bertsimas and Dunn (2017); Bertsimas et al. (2019), methods that are *optimal* in some sense. In spite of that, there is still some extra room for further improvements, in particular making classifiers more robust against perturbed datasets. The contribution in this chapter goes in that direction and it augments the catalogue of classification tree methods able to handle noise in the labels of the dataset.

We have proposed a new optimal classification tree approach able to handle label noise in the data. Two main elements support our approach: the splitting rules for the classification trees are designed to maximize the separation margin between classes and wrong labels of the training sample are detected and changed at the same time that the classifier is built. The method is encoded on a Mixed Integer Second Order Cone Optimization problem so that one can solve medium size instances by any of the nowadays available off-the-shelf solvers. We report intensive computational experiments on a battery of datasets taken from UCI Machine Learning repository showing the effectiveness and usefulness of our approach.

Chapter 6

Multiclass Optimal Classification Trees with SVM-splits

In this chapter we present a novel mathematical optimization-based methodology to construct tree-shaped classification rules for multiclass instances. Our approach consists of building Classification Trees in which, except for the leaf nodes, the labels are temporarily left out and grouped into two classes by means of a SVM separating hyperplane. We provide a Mixed Integer Non Linear Programming formulation for the problem and report the results of an extended battery of computational experiments to assess the performance of our proposal with respect to other benchmarking classification methods.

6.1 Introduction

In this chapter we provide a novel multiclass classification method which is constructed using one of the most interpretable classification methods, Classification Trees, but combined with Support Vector Machines, which provide highly predictive models.

We have developed a Mathematical Programming model that allows to construct an Optimal Classification Tree for a given training sample, in which each split is generated by means of a SVM-based hyperplane. When building the tree, the labels of the observations are ignored in the branch nodes, and they are only accounted for in the leaf nodes where misclassification errors are considered. The classification tree is constructed to minimize the complexity of the tree (assuring interpretability) and also the misclassification risk (assuring predictive power). This idea has been studied in the previous chapter for biclass classification problems, where the OCTSVM model was introduced showing that it is able to outperform CART, OCT and OCT-H in different real life datasets.

In Figure 6.1 we see a CT constructed by CART for a biclass problem with maximal depth 2. We draw the classification tree, and also in the top right corner, the partition of the feature space (in this case $\mathbb{R}^2$). As can be observed, the obtained classification is not perfect (not all leaf nodes are composed by pure classes) while in this case is not difficult to construct a CT with no classification errors. This situation is caused by the myopic construction done by the CART approach that, at each node only cares on better classification at their children, but not at the final leaf nodes, while subsequent branching decisions clearly affect the overall shape of the tree. In Figure 6.1 (right) we show a solution provided by OCT-H for the same example. One can observe that when splitting the root node (orange branches) a good local split is not obtained (the nodes contain half of the observations in different classes), however, when adding the other two splits, the final leaves only have observations of the same class, resulting in a perfect classification rule for the training sample.

On the other hand, in Figure 6.2 we show how one could construct a CT with

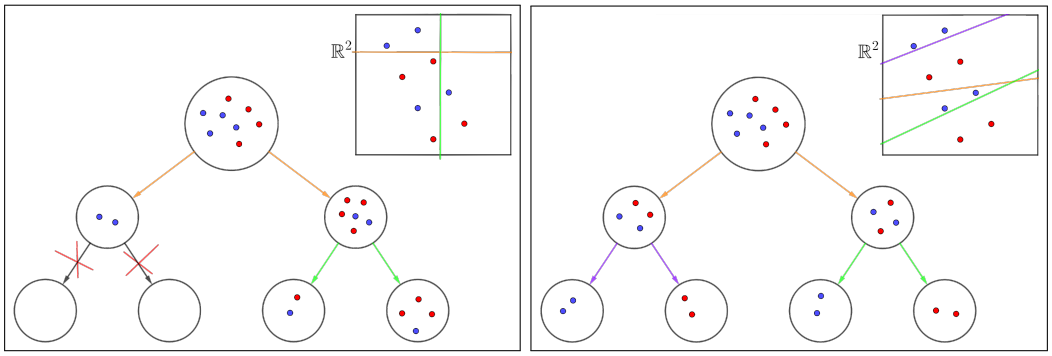


Figure 6.1: Example of a CT obtained with CART (left) and OCT-H (right) approaches for the same instance.

larger separations between the classes using OCTSVM but still with the same 100% accuracy in the training sample as in OCT-H, but more protected to misclassification in out-of-sample observations.

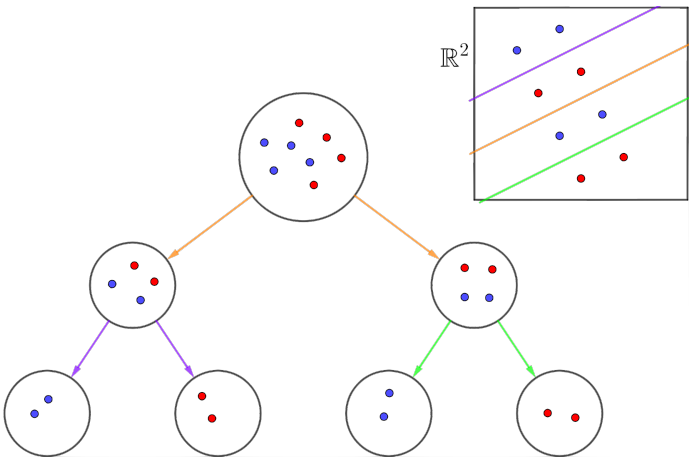


Figure 6.2: Example of a CT obtained with OCTSVM.

Nevertheless, the combination of OCT and SVM has only been analyzed for biclass instances. The extension of this method to multiclass settings (more than two classes) is not trivial, since one could construct more complex trees or use a multiclass SVM-based methodology (see e.g. Crammer and Singer (2001); Weston and Watkins (1998); Lee et al. (2004)). However, these adaptations of the classical SVM method have been proved to fail in real-world instances (see Blanco et al. (2020b)). In the rest of the chapter we describe a novel methodology to construct accurate multiclass tree-shaped classifiers based on a different idea: constructing CT with splits induced by bi-class SVM separators in which the classes of the observations at each one of the branch nodes are determined by the model, but adequately chosen to provide

small classification errors at the leaf nodes. The details of the approach are given in the following sections.

6.2 Multiclass OCT with SVM splits

In this section we describe the method that we propose to construct classification rules for multiclass instances, in particular Classification Trees in which splits are generated based on the SVM paradigm.

As already mentioned, our method, namely Multiclass Optimal Classification Trees with SVM-splits (MOCTSVM), is based on constructing OCT with SVM splits, but where the classes of the observations are momentarily ignored and only accounted for at the leaf nodes. In order to illustrate the idea under our method, in Figure 6.3 we show a toy instance with a set of points with four different classes (blue, red, orange and green).

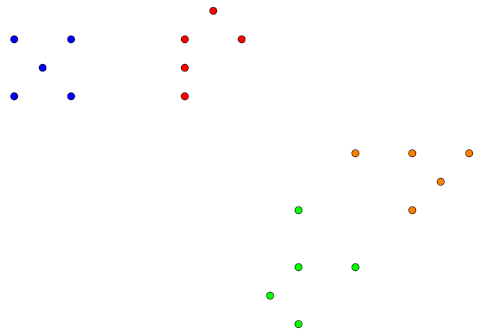


Figure 6.3: Instance for a 4-class problem.

First, at the root node (the one in which all the observations are involved), our method constructs a SVM separating hyperplane for two fictitious classes (which have to be also decided). A possible separation could be the one shown in Figure 6.4, in which the training dataset has been classified into two classes (black and white). This separation allows one to generate two child nodes, the black and the white nodes. At each of these nodes, the same idea is applied until the leaf nodes are reached. In Figure 6.5 we show the final partition of the feature space according to this procedure.

Clearly, ignoring the original classes of the training sample in the whole process would result in senseless trees, unless one accounts for the goodness in the classification rule in the training sample at the leaf nodes. Thus, at the final leaf nodes, the original labels are recovered and the classification is performed according to the generated hyperplanes. The final result of this tree is shown in Figure 6.6 where one

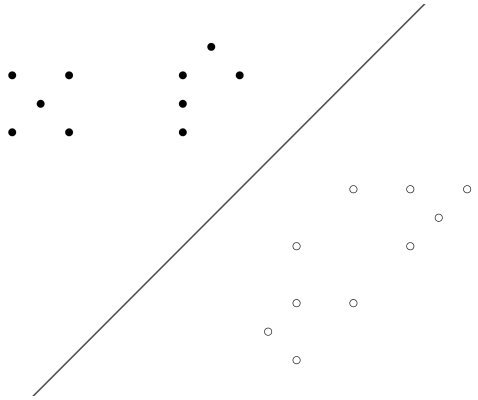


Figure 6.4: Root split on the 4-class classification problem

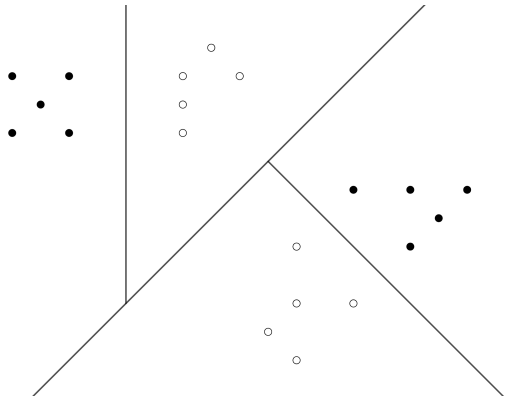


Figure 6.5: Child node splits on the 4-class classification problem higheleted as the fictitious classes decided in our model.

can check that the constructed tree achieves a perfect classification of the training sample.

Once the tree is constructed with this strategy, the decision rule comes up naturally as it is usually done in decision trees methods, that is, out of sample observations will follow a path on the tree according to the splits and they will be assigned to the class of the leaf where they lie in (the most represented class of the leaf over the training set). In case a branch is pruned when building the tree, observations will be assigned to the most represented class of the node where the prune took place.

6.3 Mathematical Programming formulation for MOCTSVM

In this section we derive a Mixed Integer Non Linear Programming formulation for the MOCTSVM method described in the previous section.

We assume to be given a training sample $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times$

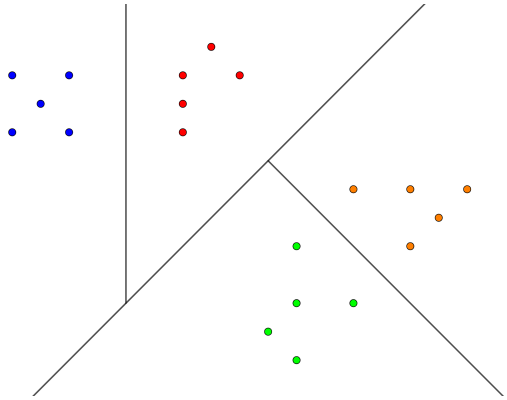


Figure 6.6: Child node splits on the 4-class classification problem with their original labels (colors).

$\{1, \ldots, k\}$. We also consider the binary representation of the labels y as:

$$Y_{ik} = \begin{cases} 1 & \text{if } y_i = s, \\ 0 & \text{otherwise,} \end{cases} \quad \text{for all } i = 1, \ldots, n, \ s = 1, \ldots k.$$

Moreover, without loss of generality we will assume the features to be normalized, i.e., $x_1, \ldots, x_n \in [0, 1]^p$.

We will construct decision trees with a fixed maximum depth D. Thus, the classification tree is formed by at most $T = 2^{D+1} - 1$ nodes. We denote by $\tau = \{1, \ldots, T\}$ the index set for the tree nodes, where node 1 is the root node and nodes $2^D, \ldots, 2^{D+1} - 1$ are the leaf nodes.

For any node $t \in \tau \backslash \{1\}$, we denote by $p(t)$ its (unique) parent node. The tree nodes can be classified in two sets: branching and leaf nodes. The branching nodes, that we denote by τ_b, will be those in which the splits are applied. In constrast, in the leaf nodes, denoted by τ_l, no splits are applied but is where predictions take place. The branching nodes can be also classified into two sets: τ_{bl} and τ_{br} depending on whether they follow the left or the right branch on the path from their parent nodes, respectively. τ_{bl} nodes are indexed with even numbers meanwhile τ_{br} nodes are indexed with odd numbers.

Moreover, as it has been done in previous chapter, we define a level as a set of nodes which have the same depth within the tree. The number of levels in the tree to be built is $D + 1$ since the root node is assumed as the zero-level. Let $U = \{u_0, \ldots, u_D\}$ be the set of levels of the tree, where each $u_r \in U$ is the set of nodes at level r, for $r = 0, \ldots, D$. With this notation, the root node is u_0 while u_D represent the set of leaf nodes.

In Figure 6.7 we show the above mentioned elements in a 3-depth tree.

Apart from the information about the topological structure of the tree, we also

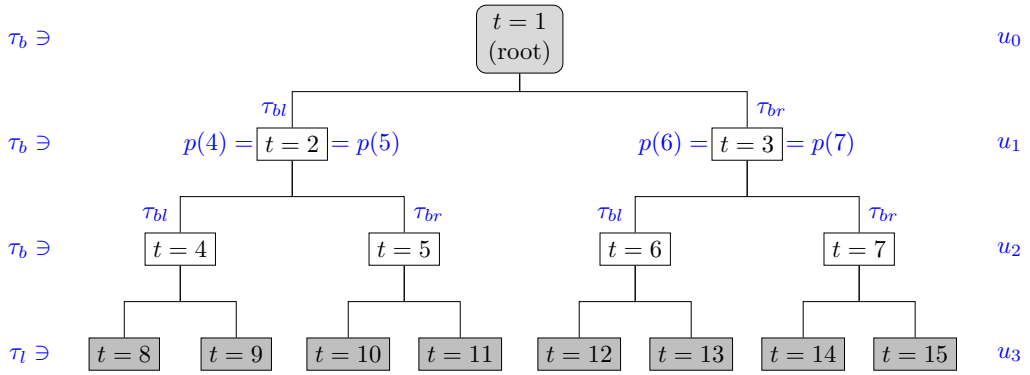


Figure 6.7: Elements in a depth $D = 3$ tree.

consider three regularization parameters that have to be calibrated in the validation process that allow us to find a trade-off between the different goals that we combine in our model: margin violation and classification errors of the separating splitting hyperplanes, correct classification at the leaf nodes and complexity of the tree. These parameters are the following:

c_1: unit misclassification cost at the leaf nodes.

c_2: unit distance based misclassification errors for SVM splits.

c_3: unit cost for each splitting hyperplane introduced in the tree.

Our model uses a set of decision and auxiliary variables that are described in Table 6.1. We use both binary and continuous decision variables to model the MOCTSVM. The binary variables allow us to decide the allocation of observations to the decision tree nodes, or to decide whether a node is splited or not in the tree. The continuous variables allow us to determine the coefficients of the splitting hyperplanes or the misclassification errors (both in th SVM separations or at the leaf nodes). We also use auxiliary binary and integer variables that are useful to model adequately the problem.

In Figure 6.8 we illustrate the use of these variables in a feasible solution of a toy instance with three classes (red,blue and green).

The whole set of training observation is considered at the root node (node $t = 1$). There, the original labels are ignored and to determine the *fictitious* class of each observation a SVM-based hyperplane is constructed. Such a hyperplane is defined by the coefficients $\omega_1 \in \mathbb{R}^p$ and $\omega_{01} \in \mathbb{R}$ (hyperplane/line drawn with a dotted line in the picture) and it induces a margin separation ($\frac{2}{\|\omega_1\|_2}$) and misclassification errors e_{i1}. In the feasible solution drawn in the figure, only three observations induce positive errors (those that are classified either in the margin area or in the opposite side of the hyperplane). Such a hyperplane also determines the splitting rule for

Continuous Decision Variables	
$\omega_t \in \mathbb{R}^p$	Coefficients of the separating hyperplane of node t.
$\omega_{t0} \in \mathbb{R}$	Intercept of the separating hyperplane of node t.
$e_{it} \in \mathbb{R}_+$	Misclassification error of observation i at node t.
$\iota \in \mathbb{R}_+$	Inverse of the minimum margin between splitting hyperplanes.
Binary Decision Variables	
$z_{it} \in \{0,1\}$	Is one if observation i belongs to node t and zero otherwise.
$\delta_t \in \{0,1\}$	Is one if a split is applied at node t and zero otherwise.
Auxiliary Variables	
$L_t \in \mathbb{Z}_+$	Number of misclassified observations at leaf node t.
$\alpha_{it} \in \{0,1\}$	Is one if observation i belongs to the reference fictitious class in node t and zero otherwise.
$h_{it} \in \{0,1\}$	Is one if observation i is in node t and lies on the positive half space of the hyperplane of node t, and zero otherwise.
$v_t \in \{0,1\}$	Is one if not all observations in node t lie on the positive half space of the hyperplane in node t and zero otherwise.
$q_{st} \in \{0,1\}$	Is one if class s is the most represented one in leaf node t and zero otherwise.

Table 6.1: Summary of the variables used in our model.

the definition of the children of that node. Since the node is split ($\delta_1 = 1$), the observations that belongs to the *positive side* of the hyperplane are assigned to the left node (node $t = 2$) while those in the negative side are assigned to the right node (node $t = 3$) through the z-variables. At node $t = 2$, the same scheme is applied, that is, the hyperplane defined by ω_2 is constructed, inducing SVM-based margin and errors and since $\delta_2 = 1$, also the splitting rule applies to define nodes $t = 4$ and $t = 5$. At node $t = 2$, one must control the observations in that node to quantify the misclassifying errors, e_{i2}, only for those observations in the objective function. Specifically, we only account for these errors for the observations that belong to the node ($z_{i2} = 1$) and either belong to the positive ($\alpha_{i2} = 1$) or the negative ($\alpha_{i2} = 0$) side of the hyperplane. Also, in order to control the complexity of the tree, the h-variables are used to know whether an observation belongs to the node and to the positive side of the SVM-hyperplane. If all observations in a node belong to the positive side of the hyperplane, the variable v assumes the value 0. Otherwise, in case v takes value 1, two situations are possible: 1) there are observations in both sides of the hyperplane (as in node $t = 2$) inducing a new split ($\delta_2 = 1$), and 2) all

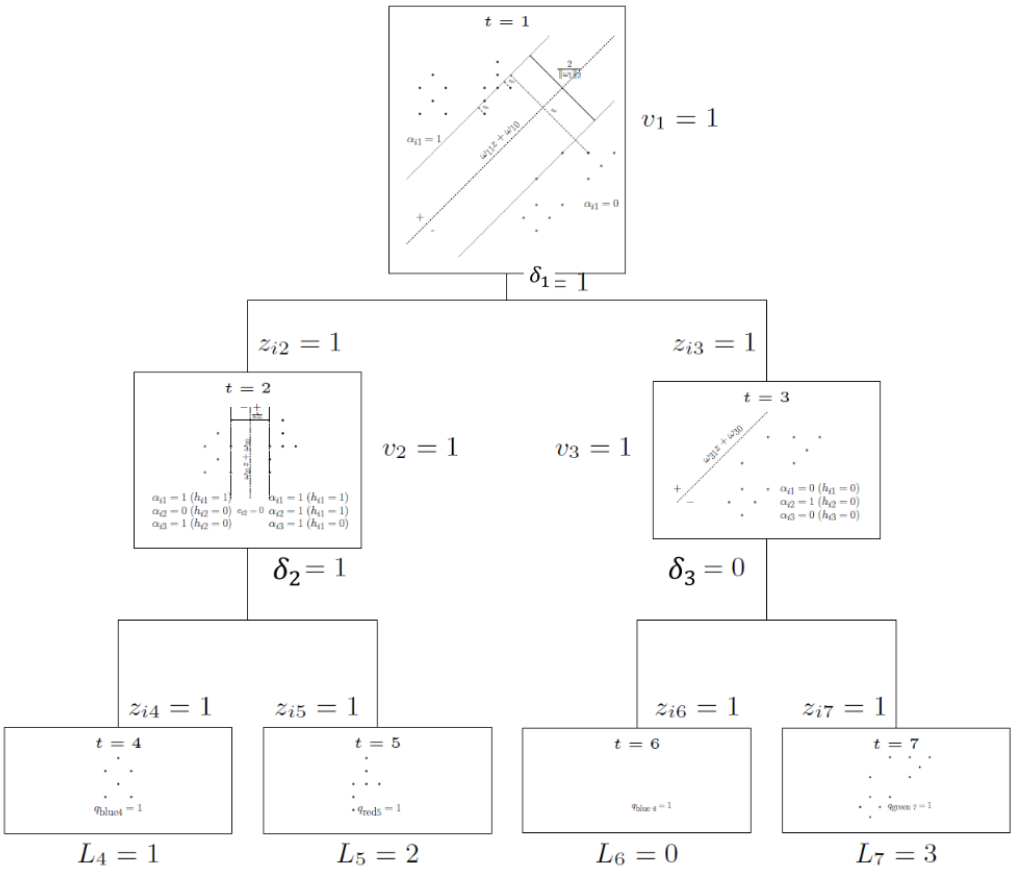


Figure 6.8: Illustration of the sets of variables used in our model in a toy example.

observations belong to the negative side (as in node $t = 3$) determining that the tree is pruned at that node ($\delta_3 = 0$).

Concerning the leaf nodes, node $t = 2$ is split into nodes $t = 4$ and $t = 5$ and node $t = 3$, which was decided to be no longer split, is fictitiously split in two leaf nodes, although one of them is empty and the other one receives all the observations of the parent node (node $t = 3$). The allocation of any leaf node τ_l to a class is done through the q-variables (to the most popular class in the node or arbitrarily in case the node has no observations) and the number of misclassified observations is accounted for by the L-variables.

As already mentioned, our method aims to construct classification trees with small misclassification errors at the leaf nodes, but at the same time with maximal separation between the classes with the SVM-based hyperplanes and minimum distance based errors. Using the variables described above, the four terms that are included in the objective functions are the following:

Margins of the splitting hyperplanes: The separating hyperplane of branching node $t \in \tau_b$ has margin $\frac{2}{\|\omega_t\|_2}$. Thus, our method aims to maximize the min-

imum of these margins. This is equivalent to minimize the maximum among the inverse margins $\{\frac{1}{2}\|\omega_t\|_2^2 : t \in \tau_b\}$ which is represented by the auxiliary variable ι.

Misclassification Errors at the leaf nodes: Variable L_t accounts for the number of misclassified observations in leaf node t, i.e., the number of observations that do not belong to the most represented class in that leaf node. These variables allow us to count the overall number of misclassified observations in the training sample. Therefore, the amount to be minimized by the model is given by the following sum:

$$c_1 \sum_{t \in \tau_l} L_t$$

Distance-based Errors at branching nodes: Each time a split is added to the tree, a SVM-based hyperplane in which the labels are assigned based on the global convenience of for the overall tree is incorporated. Thus, we measure, at each branching node in τ_b, the distance-based errors incurred by the SVM classifier at that split. This amount is measured by the e_{it} variables and is incorporated to the model through the sum:

$$c_2 \sum_{i=1}^{n} \sum_{t \in \tau_b} e_{it}$$

Complexity of the tree The simplicity of the resulting tree is measured by the number of splits that are done in its construction. Since the δ_t variable tells us whether node t is split or not, this term is accounted for in our model as:

$$c_3 \sum_{t \in \tau_b} \delta_t$$

Summarizing, the overall objective function of our model is:

$$\min \ \iota + c_1 \sum_{t \in \tau_l} L_t + c_2 \sum_{i=1}^{n} \sum_{t \in \tau_b} e_{it} + c_3 \sum_{t \in \tau_b} \delta_t. \tag{OBJ}$$

Note that the coefficients c_1, c_2 and c_3 trade-off the misclassification of the training sample, the separation between classes and the complexity of the tree, respectively. These parameters should be carefully calibrated in order to construct *simple* decision trees with high predictive power, as can be seen in our computational experiments.

The requirements on the relationships between the variables and the rationale of our model are described through the following constraints that define the Math-

ematical Programming model.
First of all, in order to adequately represent the maximum among the inverse margins of the sppliting hyperplanes, we require:

$$\iota \geq \frac{1}{2}\|\omega_t\|_2^2, \qquad \forall t \in \tau_b. \tag{6.1}$$

Next, we impose how the splits are performed in the tree. To this end, we need to know which observations belong to a certain node t (z-variable) and how these observations are distributed with respect to the two fictitious classes to be separated (α-variables). Gathering all these elements together, we use the following constraints to define the splits of the decision tree:

$$\omega_t' x_i + \omega_{t0} \geq 1 - e_{it} - (1 - h_{it}), \qquad \forall i = 1, \ldots, n,\ t \in \tau_b, \tag{6.2}$$

$$\omega_t' x_i + \omega_{t0} \leq -1 + e_{it} + (1 - z_{it} + \alpha_{it}), \qquad \forall i = 1, \ldots, n,\ t \in \tau_b. \tag{6.3}$$

According to this, constraint (6.2) is activated just in case the observation i belongs to the reference class and it is in node t ($h_{it} = 1$). On the other hand, (6.3) is activated if i is allocated to node t ($z_{it} = 1$) but it does not belong to the reference class ($\alpha_{it} = 0$). Therefore, the reference class is located on the positive half space of hyperplane $\mathcal{H}_t$, while the other class is positioned in the negative half space, and at the same time, margin violations are regulated by the e_{it} variables.

To ensure the correct behaviour of the above constraints, we must correctly define the z_{it} variables. First, it is required that each observation belongs to exactly one node per level in the tree. This can be easily done by adding the usual assingment constraints to the problem at each of the levels, $u \in U$, of the tree:

$$\sum_{t \in u} z_{it} = 1, \qquad \forall i = 1, \ldots, n,\ u \in U. \tag{6.4}$$

Furthermore, we should enforce that if observation i is in node t ($z_{it} = 1$), then observation i must also be in the parent node of t, $p(t)$ ($z_{ip(t)} = 1$), and also observation i can not be in node t if it is not in its parent node ($z_{ip(t)} = 0 \Rightarrow z_{it} = 0$). These implications can be obtained by means of the following constraints:

$$z_{it} \leq z_{ip(t)}, \qquad \forall i = 1, \ldots, n,\ t = 2, \ldots, T. \tag{6.5}$$

Nevertheless, the way observations descend through the tree needs a further analysis, since at this point they could just randomly define a path in the tree. Whenever an observation i is in the positive half space of the splitting hyperplane at node t, $\mathcal{H}_t$, this observation should follow the right branch connecting to the child node of t. Otherwise, in case i is on the negative half space, it should follow the left branch.

The knowledge on the side of the splitting hyperplane where an observation belongs to is encoded in the α-variables. Then, in case i lies on the positive half space of $\mathcal{H}_t$, α_{it} will never be equal to zero since it would lead to a value of e_{it} greater than one, while $e_{it} < 1$ is guaranteed in case $\alpha_{it} = 1$.

With the above observations, the constraints that assure the correct construction of the splitting hyperplanes with respect to the side of them where the observations belong to are the following:

$$z_{ip(t)} - z_{it} \leq \alpha_{ip(t)} \qquad \forall i = 1, \ldots, n,\ t \in \tau_{bl}, \tag{6.6}$$

$$z_{ip(t)} - z_{it} \leq 1 - \alpha_{ip(t)} \qquad \forall i = 1, \ldots, n,\ t \in \tau_{br}. \tag{6.7}$$

Constraints (6.6) assure that if observation i is on the parent node of an even node t ($z_{ip(t)} = 1$), and i lies on the negative half space of $\mathcal{H}_{p(t)}$ ($\alpha_{ip(t)} = 0$), then z_{it} is enforced to be equal to one. As a result, $\alpha_{ip(t)} = 0$ forces observation i to take the left branch in node t. Note that in case $z_{ip(t)} = 1$, and at the same time observation i is not in the left child node of t ($z_{it} = 0$ for $i \in \tau_{bl}$), then $\alpha_{ip(t)} = 1$, which means that observation i lies on the positive half space of $\mathcal{H}_{p(t)}$. Constraints (6.7) are analogous to (6.6) but allowing to adequately represent right branching nodes.

Moreover, two additional important elements need to be incorporated to complete our model: the tree complexity and the correct definition of misclassified observations. Recall that OCT and OCT-H do not use SVM-based splits, and therefore the complexity can be easily regulated by just imposing $\|\omega_t\|_2^2 \leq M\delta_t$ (for a big enough M constant) in all the branch nodes, since in case a node is no further branched ($\delta_t = 0$), the coefficients of the splitting hyperplane are set to zero. However, in our case, in which the splitting hyperplanes are SVM-based hyperplanes, these constraints are in conflict with constraints (6.2) and (6.3), since in case $\delta_t = 0$ (and therefore $\omega_t = 0$) it would not only imply that the coefficients ω_t are equal to zero, but also that the distance based errors would be set to the maximum value of 1, i.e., $e_{it} = 1$ for every observation i in the node, even though these errors would not make any sense since observations would not be separated at the node. To overcome this issue, we consider the auxiliary binary variables $h_{it} = z_{it}\alpha_{it}$ (h_{it} takes value 1 if observation i belongs to node t and lies in the positive half-space of the splitting hyperplane applied at node t) and v_t (that takes value zero in case all the points in the node belong to the positive halfspace and one otherwise). The variables are

adequatelly defined if the following constraints are incorporated to the model:

$$h_{it} \geq z_{it} + \alpha_{it} - 1, \qquad \forall i = 1, \ldots, n \; t \in \tau_b, \tag{6.8}$$

$$h_{it} \leq z_{it} - \alpha_{it} + 1, \qquad \forall i = 1, \ldots, n \; t \in \tau_b, \tag{6.9}$$

$$\sum_{i=1}^{n} (z_{it} - h_{it}) \leq n v_t, \qquad \forall t \in \tau_b, \tag{6.10}$$

$$\sum_{i=1}^{n} h_{it} \leq n(1 + \delta_t - v_t), \qquad \forall t \in \tau_b, \tag{6.11}$$

where constraints (6.8) and (6.9) are the linearization of the bilinear constraint $h_{it} = z_{it}\alpha_{it}$. On the other hand, Constraints (6.10) assure that in case $v_t = 0$, then all observations in node t belong to the positive halfspace of $\mathcal{H}_t$, and constraints (6.11) assure that if $v_t = 1$ and the tree is pruned at node t ($\delta_t = 0$), then those observations allocated to node t are placed in the negative halfspace defined by the splitting hyperplane. Thus, it implies that δ_t takes value one if and only if the observations in node t are separated by $\mathcal{H}_t$, and therefore producing an effective split at the node.

Finally, in order to adequately represent the L_t variables (the ones that measure the number of misclassified observations at the leaf nodes) we use the constraints already incorporated in the OCT-H model in Bertsimas and Dunn (2017). On the one hand, we assign each leaf node to a single class (the most popular class of the observations that belong to that node). We use the binary variable q_{st} to check whether leaf node $t \in \tau_l$ is assigned to class $s = 1, \ldots, k$. The usual assignment constraints are considered to assure that each node is assigned to exactly one class:

$$\sum_{s=1}^{k} q_{st} = 1, \qquad \forall t \in \tau_l. \tag{6.12}$$

The correct definition of the variable L_t is then guaranteed by the following set of constraints:

$$L_t \geq \sum_{i=1}^{n} z_{it} - \sum_{i=1}^{n} Y_{is} z_{it} - n(1 - q_{st}), \qquad \forall s = 1, \ldots, k, \; t \in \tau_l. \tag{6.13}$$

These constraints are activated if and only if $q_{st} = 1$, i.e., if observations in node t are assigned to class s. In such a case, since L_t is being minimized in the objective function, L_t will be determined by the number of training observations in node t except those whose label is s, i.e., the number of missclasified observations in node t according to the s-class assignment.

Observe that the constant n in (6.13) can be decreased and fixed to the maximum

number of misclassified observations in the training sample. This number coincide with the difference between the number of observations in the training sample (n) and the number of observations in the most represented class in the sample.

Summarizing the above paragraphs, the MOCTSVM can be formulated as the following MINLP problem:

$$
\begin{aligned}
\min \quad & \iota + c_1 \sum_{t\in\tau_l} L_t + c_2 \sum_{i=1}^{n}\sum_{t\in\tau} e_{it} + c_3 \sum_{t\in\tau} \delta_t && && \text{(MOCTSVM)}\\
\text{s.t.} \quad & \iota \geq \frac{1}{2}\|\omega_t\|, && && \forall t \in \tau_b,\\
& \omega_t' x_i + \omega_{t0} \geq 1 - e_{it} - (2 - z_{it} - \alpha_{it}), && && \forall i = 1,\dots,n,\ t \in \tau_b,\\
& \omega_t' x_i + \omega_{t0} \leq -1 + e_{it} + (1 - z_{it} + \alpha_{it}), && && \forall i = 1,\dots,n, t \in \tau_b,\\
& \sum_{t\in u} z_{it} = 1, && && \forall i = 1,\dots,n,\ u \in U,\\
& z_{it} \leq z_{ip(t)}, && && \forall i = 1,\dots,n,\ t = 2,\dots,T,\\
& z_{ip(t)} - z_{it} \leq \alpha_{ip(t)}, && && \forall i = 1,\dots,n,\ t \in \tau_{bl},\\
& z_{ip(t)} - z_{it} \leq 1 - \alpha_{ip(t)}, && && \forall i = 1,\dots,n,\ t \in \tau_{br},\\
& h_{it} \geq z_{it} + \alpha_{it} - 1, && && \forall i = 1,\dots,n,\ t \in \tau_b,\\
& h_{it} \leq z_{it} - \alpha_{it} + 1, && && \forall i = 1,\dots,n,\ t \in \tau_b,\\
& \sum_{i=1}^{n} (z_{it} - h_{it}) \leq n v_t, && && \forall t \in \tau_b,\\
& \sum_{i=1}^{n} h_{it} \leq n(1 + \delta_t - v_t), && && \forall t \in \tau_b,\\
& \sum_{s=1}^{k} q_{st} = 1, && && \forall t \in \tau_l,\\
& L_t \geq \sum_{i=1}^{n} z_{it} - \sum_{i=1}^{n} Y_{is} z_{it} - n(1 - q_{st}), && && \forall s = 1,\dots,k,\ t \in \tau_l,\\
& e_{it} \in \mathbb{R}^+, \alpha_{it}, h_{it} \in \{0,1\}, && && \forall i = 1,\dots,n,\ t \in \tau_b,\\
& z_{it} \in \{0,1\}, && && \forall i = 1,\dots,n,\ t = 1,\dots,T,\\
& q_{st} \in \{0,1\}, && && \forall s = 1\dots,k,\ t \in \tau_l,\\
& \omega_t \in \mathbb{R}^p, \omega_{t0} \in \mathbb{R}, \delta_t \in \{0,1\}, && && \forall t = 1,\dots,T.
\end{aligned}
$$

6.3.1 Strengthening the model

The MINLP formulation presented above is valid for our MOCTSVM model. However, it is a computationally costly problem, and although it can be solved by most of the off-the-shelf optimization solvers, it is able to solve optimally only small to

medium size instances. To improve its performance, the problem can be strengthen by means of valid inequalities which allows one to reduce the gap between the continuous relaxation of the problem and its optimal integer solution, being then able to solve larger instances in smaller CPU times. In what follows we describe some of these inequalities that we have incorporated to the MINLP formulation:

- If observations i and i' belongs to different nodes, they cannot be assigned to the same node for the remainder levels of the tree:

$$z_{ir} + z_{i'r} \leq z_{it} + z_{i't}, \quad \forall t \in u,\ r \in u'u \leq u'$$

- If leaf nodes t and t' are the result of proper splitting hyperplanes, then, both nodes cannot be assigned to the same class:

$$q_{st} + q_{st'} \leq 2 - d_{p(t)}, \quad \forall t, t' = 2, \ldots, T(t \neq t') \text{ with } p(t) = p(t'),\ s = 1, \ldots, k.$$

- Variable α_{it} is enforced to take value 0 in case $z_{it} = 0$:

$$\alpha_{it} \leq z_{it}, \quad \forall i = 1, \ldots, n,\ t \in \tau_b.$$

- Variable h_{it} is not allowed to take value one if α_{it} takes value zero:

$$h_{it} \leq \alpha_{it}, \quad \forall i = 1, \ldots, n,\ t \in \tau_b.$$

- There should be at least a leaf node to which each class is assigned to (assuming that each class is represented in the training sample). It also implies that the number of nodes to which a class is assigned is bounded as:

$$1 \leq \sum_{t \in \tau_l} q_{st} \leq 2^D - 1, \quad \forall s = 1, \ldots, k.$$

6.4 Experiments

In order to analyze the performance of this new methodology we have run a series of experiments among different real datasets from UCI Machine Learning Repository Lichman et al. (2013). We have chosen twelve datasets with number of classes between two and seven. The dimension of these problems is reported in Table 6.2 by the tuple (n : number of observations, p : number of features, k : number of classes).

We have compared the MOCTSVM model with three other Classification Tree-based methodologies, namely CART, OCT and OCT-H. The maximum tree depth, D, for all the models was equal to 3, and the minimum number of observations per

node in CART, OCT and OCT-H was equal to the 5% of the training size.

We have performed, for each instance a 5-fold cross validation scheme, i.e., datasets have been splited into five random train-test partitions where one of the folds is used to build the model and the remaining are used to measure the accuracy of the predictions. Moreover, in order to avoid taking advantage of beneficial initial partitions, we have repeated the cross-validation scheme five times for all the datasets.

The CART method was coded in `R` using the `rpart` library. On the other hand, MOCTSVM, OCT and OCT-H were coded in `Python` and solved using the optimization solver `Gurobi` 8.1.1. All the experiments were run on a PC Intel Xeon E-2146G processor at 3.50GHz and 64GB of RAM. A time limit of 300 seconds was set for training the training folds. Although not all the problems were optimally solved within the time limit, as can be observed in Table 6.2, the results obtained with our model already outperform the other methods.

In order to calibrate the parameters of the different models that regulate the complexity of the tree, we have used different approaches. On the one hand, for CART and OCT, since the maximum number of nodes for such a depth is $2^D - 1 = 7$, one can search for the tree with best complexity by searching in the grid $\{1, \ldots, 2^D - 1\}$ of possible active nodes. For OCT-H, we search the complexity regularization factor in the grid $\{10^i : i = -5, \ldots, 5\}$. Finally, in MOCTSVM we used the same grid $\{10^i : i = -5, \ldots, 5\}$ for c_1 and c_2, and $\{10^i : i = -2, \ldots, 2\}$ for c_3.

In Table 6.2 we report the results obtained in our experiments for all the models. The first column of the table indicates the identification of the dataset (together with its dimensionality). Second, for each of the methods that we have tested, we report the obtained average test accuracy and the standard deviation. We have highlighted in bold the best average test accuracies obtained for each dataset.

As can be observed, our method clearly outperforms in most of the instances the rest of the methods in terms of accuracy. Clearly, our model is designed to construct Optimal Classification Trees with larger separations between the classes, which results in better accuracies in the test sample. The datasets *Australian* and *BalanceScale* obtain their better results with OCT-H, but, as can be observed, the differences with respect the rest of the methods are tiny (it is the result of correctly classifying in the test sample just a few more observations than the rest of the methods). In that case, our method gets an accuracy almost as good as OCT-H. In the rest of the datasets, our method consistently gets better classifiers and for instance for *Dermatology* the difference with respect to the best classifiers among the others ranges in $[4\%, 19\%]$, for *Parkinson* the accuracy with our model is at least 6% better than the rest, for Wine we get 5% more accuracy than OCTH and 10%

more than CART and for Zoo the accuracy of our model is more than 17% greater than the one obtained with CART.

Concerning the variability of our method, the standard deviations reported in Table 6.2 show that our results are, in average, more stable than the others, with small deviations with respect to the average accuracies. This behaviour differs from the one observed in CART or OCT, where larger deviations are obtained, implying that the accuracies highly depends of the test folder where the method is applied.

	CART	OCT	OCT-H	MOCTSVM	Diff
Australian (690,14,2)	85.54 ± 0.81	85.22 ± 1.27	**85.65 ± 1.02**	85.27 ± 1.11	-0.38 ± 0.63
BalanceScale (625,4,3)	69.55 ± 1.76	73.30 ± 1.20	**90.43 ± 1.07**	89.53 ± 1.28	-0.90 ± 1.69
Banknote (1372,5,2)	89.27 ± 0.95	88.50 ± 1.17	98.89 ± 0.33	**98.91 ± 0.46**	0.02 ± 0.43
BreastCancer (683,9,2)	92.69 ± 1.01	94.16 ± 0.54	95.10 ± 1.26	**96.27 ± 0.64**	1.17 ± 1.39
Dermatology (358,34,6)	75.69 ± 3.60	77.82 ± 4.34	91.41 ± 2.83	**95.39 ± 1.47**	3.98 ± 2.74
Heart (294,13,5)	64.37 ± 1.48	65.14 ± 1.57	64.30 ± 1.79	**66.41 ± 1.54**	1.26 ± 1.38
Iris (150,4,3)	94.26 ± 1.90	95.37 ± 0.97	95.64 ± 1.46	**95.72 ± 1.79**	0.08 ± 1.70
Parkinson (240,40,2)	72.29 ± 4.05	73.53 ± 2.26	74.92 ± 3.01	**80.83 ± 1.89**	5.91 ± 3.06
Seeds (210,7,3)	86.36 ± 4.02	88.52 ± 2.69	91.12 ± 2.99	**92.98 ± 1.82**	1.85 ± 2.23
Teaching (150,5,3)	41.91 ± 5.64	48.35 ± 3.80	48.09 ± 2.92	**48.62 ± 3.37**	0.26 ± 4.60
Thyroid (215,5,3)	89.77 ± 2.37	92.43 ± 2.12	92.46 ± 2.49	**94.57 ± 2.08**	2.11 ± 3.10
Wine (178,13,3)	84.52 ± 2.66	92.22 ± 3.41	89.35 ± 3.71	**94.13 ± 1.78**	1.90 ± 3.37
Zoo (101,16,7)	74.96 ± 5.79	87.75 ± 1.99	89.11 ± 2.58	**92.31 ± 2.15**	3.20 ± 2.95

Table 6.2: Average accuracies (± standard deviations) obtained in our computational experiments.

6.5 Conclusions

We have presented in this chapter a novel methodology to construct classifiers for multiclass instances by means of a Mathematical Programming model. The proposed method outputs a classification tree were the splits are based on SVM-based hyperplanes. At each branch node of the tree, a binary SVM hyperplane is constructed in which the observations are classified in two fictitious classes (the original classes are ignored in all the splitting nodes), but the global goodness of the tree is measured at the leaf nodes, where misclassification errors are minimized. Also, the model minimizes the complexity of the tree together with the two elements that appear in SVM-approaches: margin separation and distance-based misclassifying errors. We have run an extensive battery of computational experiments that shows that our method outperforms most of the Decision Tree-based methodologies both in accuracy and stability.

Chapter 7

Conclusions and future research lines

Motivated by the advances made in Mathematical Programming over Data Science in recent years, in this thesis we have focused on analyzing hyperplanes location to address Classification and Aggregation problems. Throughout all the chapters of this dissertation we have developed MILP and MINLP formulations to design new models in which theoretical properties have been discussed along with an extensive computational analysis to measure the performance of the proposed methodologies, thus proving their validity. In the following, we briefly summarize the major achievements and discuss possible future research lines of each chapter.

Chapter 2 proposes a novel modeling framework based on the Support Vector Machines to address multiclass classification problems. In such a framework, the classes are linearly separated and the separation between classes is maximized. Different approaches are presented regarding to two kinds of error and distance measures. These approaches compute an optimal arrangement of hyperplanes subdividing the space into cells, and so that each cell is assigned to a class based on the training sample. Dimensionality reduction techniques are developed in order to help solvers finding the optimal solutions of the problems. Furthermore, we prove that an analogous of the kernel trick is valid for our methods. The performance of this approach is tested amongst a set of real and synthetic datasets, in which our methods showed to work remarkably well compared to other methods existing in the literature.

Several extensions of our approach are possible. Amongst them we would like to mention the use of heuristic algorithms to solve the complex MILP and MINLP problems which may alleviate the computational burden of the methodology still keeping high quality solutions. Moreover, our approach could also be extended to the framework of semisupervised learning by assigning unlabelled observations to their closest well-classified cells.

In Chapter 3 we study the problem of locating a set of hyperplanes in order to minimize an objective function of the distances from a set of points where each point is assigned to its closest hyperplane. The distance from each point to its corresponding fitting hyperplane can be seen as a residual, and these residuals are aggregated using ordered median functions. Two exact approaches are presented to solve the problem. One based on a compact mixed integer formulation and another one based on an extended set partitioning formulation with an exponential number of variables handled by a branch-and-price approach. We developed initial feasible solutions to initialize the column generation procedure of this branch-and-price. Moreover, we present a heuristic pricing strategy that is used in combination with the exact one to speed up the pricing iterations. In our computational study we report the comparison of both methods over two different real datasets and an extensive battery of synthetic instances. Finally, we analyze the issue of scalability of the exact methods, obtaining theoretical upper bounds of the error induced by

some aggregated versions of the original dataset.

A possible extension to be developed in future studies is the development of alternative heuristic algorithms capable to solve the problem for large instances. Other types of tools could also be explored under the multisource ordered median paradigm, as for instance, in view of the good results obtained in Chapter 2, Support Vector Machines.

Chapter 4 presents a methodology to construct a classification rule that simultaneously incorporates the detection of label noise within the datasets. This methodology combines SVM and clustering techniques to simultaneously identify wrong labels while building a separating hyperplane maximizing the margin and minimizing misclassification errors, allowing observations to be relabeled in the training process. The rationale is simple: observations identified as wrongly labeled will be relabelled only if the gain in margin or the decrease in misclassification error compensate the flipping. Our findings are not only of theoretical interest. Its practical performance when applied to real datasets is remarkable. In all tested cases, our methods are superior to the considered benchmark. Thus, they are directly applicable to datasets in which flipped labels are suspected, resulting in robust classifiers to noisy labels.

Further research on the topic includes the extension of our models to deal with multiclass instances by modifying the *relabel*-variables to identify the new (non-binary) labels. The strategy should be carefully chosen using a multiclass SVM-based approach (as One versus One, One versus All or any of the unified tools). This extension is not trivial and requires a deeper analysis.

Other lines of research that would extend our methods are the application of alternative clustering strategies, as those based on ordered median objective functions or the twin SVM methodology. Also, the use of kernel tools in our approaches, in order to be able to construct non linear classifiers has to be investigated.

Chapters 5 and 6 approach the problem of building optimal Classification Trees with SVM-based splits. On the one hand, Chapter 5 introduces the OCTSVM model, which is designed to deal with binary classification problems and which is able to handle label noise in the data. On the other hand, Chapter 6 presents the MOCTSVM, a model that extends the OCTSVM to the multiclass scenario. We have reported intensive computational experiments on a battery of datasets taken from UCI Machine Learning repository, showing that our methods outperform most of the Decision Tree-based methodologies both in accuracy and stability.

Future research lines on this topic include the analysis of nonlinear splits when branching in OCTSVM and MOCTSVM, both using kernel tools derived from SVM classifiers or specific families of nonlinear separators. This approach will result into more flexible classifiers able to capture the nonlinear trends of many real-life datasets. Furthermore, we also plan to incorporate features selection in our methods

in order to construct highly predictive but also more interpretable classification tools. Additionally, we also plan to address the design of math-heuristic algorithms which keep the essence of OCT but capable to train larger datasets.

References

Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., Gramfort, A., Thirion, B., and Varoquaux, G. (2014). Machine learning for neuroimaging with scikit-learn. *Frontiers in neuroinformatics*, 8:14.

Allwein, E. L., Schapire, R. E., and Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, 1(Dec):113–141.

Amaldi, E., Coniglio, S., and Taccari, L. (2016). Discrete optimization methods to fit piecewise affine models to data points. *Computers & Operations Research*, 75:214–230.

Archer, K. J. and Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. *Computational statistics & data analysis*, 52(4):2249–2260.

Bagirov, A. M., Ugon, J., Webb, D., Ozturk, G., and Kasimbeyli, R. (2013). A novel piecewise linear classifier based on polyhedral conic and max–min separabilities. *Top*, 21(1):3–24.

Bahlmann, C., Haasdonk, B., and Burkhardt, H. (2002). Online handwriting recognition with support vector machines-a kernel approach. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 49–54. IEEE.

Balas, E. and Padberg, M. W. (1976). Set partitioning: A survey. *SIAM review*, 18(4):710–760.

Baldomero-Naranjo, M., Martínez-Merino, L. I., and Rodríguez-Chía, A. M. (2020). Tightening big ms in integer programming formulations for support vector machines with ramp loss. *European Journal of Operational Research*, 286(1):84–100.

Baldomero-Naranjo, M., Martínez-Merino, L. I., and Rodríguez-Chía, A. M. (2021). A robust svm-based approach with feature selection and outliers detection for classification problems. *Expert Systems with Applications*, 178:115017.

Benati, S., Puerto, J., and Rodríguez-Chía, A. M. (2017). Clustering data that are graph connected. *European Journal of Operational Research*, 261(1):43–53.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252.

Benítez-Peña, S., Blanquero, R., Carrizosa, E., and Ramírez-Cobo, P. (2019). On support vector machines under a multiple-cost scenario. *Advances in Data Analysis and Classification*, 13(3):663–682.

Bennett, K., Demiriz, A., et al. (1999). Semi-supervised support vector machines. *Advances in Neural Information processing systems*, pages 368–374.

Bennett, K. P. and Blue, J. (1998). A support vector machine approach to decision trees. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 3, pages 2396–2401. IEEE.

Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Machine Learning*, 106(7):1039–1082.

Bertsimas, D., Dunn, J., Pawlowski, C., and Zhuo, Y. D. (2019). Robust classification. *INFORMS Journal on Optimization*, 1(1):2–34.

Bertsimas, D. and Dunn, J. W. (2019). *Machine learning under a modern optimization lens*. Dynamic Ideas LLC.

Bertsimas, D. and Mazumder, R. (2014). Least quantile regression via modern optimization. *The Annals of Statistics*, 42(6):2494–2525.

Bertsimas, D. and Shioda, R. (2007). Classification and regression via integer optimization. *Operations Research*, 55(2):252–271.

Bi, J. and Zhang, T. (2005). Support vector classification with input data uncertainty. *Advances in neural information processing systems*, 17(1):161–168.

Biggio, B., Nelson, B., and Laskov, P. (2011). Support vector machines under adversarial label noise. In *Asian conference on machine learning*, pages 97–112. PMLR.

Blanco, V., Japón, A., Ponce, D., and Puerto, J. (2021a). On the multisource hyperplanes location problem to fitting set of points. *Computers & Operations Research*, 128:105124.

Blanco, V., Japón, A., and Puerto, J. (2020a). A mathematical programming approach to binary supervised classification with label noise. *arXiv preprint arXiv:2004.10170*.

Blanco, V., Japón, A., and Puerto, J. (2020b). Optimal arrangements of hyperplanes for svm-based multiclass classification. *Advances in Data Analysis and Classification*, 14(1):175–199.

Blanco, V., Japón, A., and Puerto, J. (2021b). Multiclass optimal classification trees with svm-splits. *arXiv preprint arXiv:2111.08674*.

Blanco, V., Japón, A., and Puerto, J. (2021c). Robust optimal classification trees under noisy labels. *Advances in Data Analysis and Classification*, pages 1–25.

Blanco, V., Puerto, J., and Ben-Ali, S. E.-H. (2016). Continuous multifacility ordered median location problems. *European Journal of Operational Research*, 250(1):56–64.

Blanco, V., Puerto, J., and El-Haj Ben-Ali, S. (2014). Revisiting several problems and algorithms in continuous location with lp norms. *Computational Optimization and Applications, 58 (3), 563-595.*

Blanco, V., Puerto, J., and Rodríguez-Chía, A. M. (2020c). On ℓ_p-support vector machines and multidimensional kernels. *Journal of Machine Learning Research.*

Blanco, V., Puerto, J., and Rodriguez-Chia, A. M. (2020d). On lp-support vector machines and multidimensional kernels. *J. Mach. Learn. Res.*, 21:14–1.

Blanco, V., Puerto, J., and Salmerón, R. (2018). Locating hyperplanes to fitting set of points: A general framework. *Computers & Operations Research*, 95:172–193.

Boland, N., Domínguez-Marín, P., Nickel, S., and Puerto, J. (2006). Exact procedures for solving the discrete ordered median problem. *Computers & Operations Research*, 33(11):3270–3300.

Boucher, J.-P., Denuit, M., and Guillen, M. (2009). Number of accidents or number of claims? an approach with zero-inflated poisson models for panel data. *Journal of Risk and Insurance*, 76(4):821–846.

Bradley, P. S. and Mangasarian, O. L. (2000). K-plane clustering. *Journal of Global Optimization*, 16(1):23–32.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L. (2002). Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*, 1(58).

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and regression trees. wadsworth int. *Group*, 37(15):237–251.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (2017). *Classification and regression trees*. Routledge.

Brimberg, J., Juel, H., and Schöbel, A. (2002). Linear facility location in three dimensions—models and solution methods. *Operations Research*, 50(6):1050–1057.

Brimberg, J., Juel, H., and Schöbel, A. (2003). Properties of three-dimensional median line location models. *Annals of Operations Research*, 122(1-4):71–85.

Brooks, J. P. (2011). Support vector machines with the ramp loss and the hard margin loss. *Operations research*, 59(2):467–479.

Carbonneau, R. A., Caporossi, G., and Hansen, P. (2014). Globally optimal clusterwise regression by column generation enhanced with heuristics, sequencing and ending subset optimization. *J. Classif.*, 31(2):219–241.

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

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292.

Current, J. R. and Schilling, D. A. (1987). Elimination of source a and b errors in p-median location problems. *Geographical Analysis*, 19(2):95–110.

Current, J. R. and Schilling, D. A. (1990). Analysis of errors due to demand data aggregation in the set covering and maximal covering location problems. *Geographical Analysis*, 22(2):116–126.

de França, F. O. and Coelho, A. L. (2015). A biclustering approach for classification with mislabeled data. *Expert Systems with Applications*, 42(12):5065–5075.

Dıaz-Bánez, J., Mesa, J. A., and Schöbel, A. (2004). Continuous location of dimensional structures. *European Journal of Operational Research*, 152(1):22–44.

Dietterich, T. G. and Bakiri, G. (1994). Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286.

Drezner, Z., Steiner, S., and Wesolowsky, G. O. (2002). On the circle closest to a set of points. *Computers & Operations Research*, 29(6):637–650.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems*, 9:155–161.

Du, M., Liu, N., and Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77.

Duan, Y. and Wu, O. (2016). Learning with auxiliary less-noisy labels. *IEEE transactions on neural networks and learning systems*, 28(7):1716–1721.

Edgeworth, F. Y. (1887). On observations relating to several quantities. *Hermathena*, 6(13):279–285.

Eilon, S., Watson-Gandy, C. D. T., and Christofides, N. (1971). *Distribution management : mathematical modelling and practical analysis*. London : Griffin.

Ekambaram, R., Fefilatyev, S., Shreve, M., Kramer, K., Hall, L. O., Goldgof, D. B., and Kasturi, R. (2016). Active cleaning of label noise. *Pattern Recognition*, 51:463–480.

Espejo, I. and Rodríguez-Chía, A. M. (2011). Simultaneous location of a service facility and a rapid transit line. *Computers & operations research*, 38(2):525–538.

Frénay, B. and Verleysen, M. (2013). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Friedman, J. H. (2017). *The elements of statistical learning: Data mining, inference, and prediction*. springer open.

Ganapathiraju, A., Picone, J., et al. (2000). Support vector machines for automatic data cleanup. In *INTERSPEECH*, pages 210–213.

Gaudioso, M., Gorgone, E., Labbé, M., and Rodríguez-Chía, A. M. (2017). Lagrangian relaxation for svm feature selection. *Computers & Operations Research*, 87:137–145.

Gauss, C. F. (1877). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*, volume 7. FA Perthes.

Geoffrion, A. (1977). Objective function approximations in mathematical programming. *Mathematical Programming*, 13:23–39.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260.

Ghaddar, B. and Naoum-Sawaya, J. (2018). High dimensional data classification and feature selection using support vector machines. *European Journal of Operational Research*, 265(3):993–1004.

Giloni, A. and Padberg, M. (2002). Alternative methods of linear regression. *Mathematical and Computer Modelling*, 35(3-4):361–374.

Gini, C. (1912). Variabilità e mutabilità (variability and mutability). *Cuppini, Bologna*, 156.

Gitman, I., Chen, J., Lei, E., and Dubrawski, A. (2018). Novel prediction techniques based on clusterwise linear regression. *arXiv preprint arXiv:1804.10742*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Guermeur, Y. and Monfrini, E. (2011). A quadratic loss multi-class svm for which a radius–margin bound applies. *Informatica*, 22(1):73–96.

Günlük, O., Kalagnanam, J., Li, M., Menickelly, M., and Scheinberg, K. (2021). Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, pages 1–28.

Gupta, O. K. and Ravindran, A. (1985). Branch and bound experiments in convex nonlinear integer programming. *Management science*, 31(12):1533–1546.

Hampel, F. R. (1975). Beyond location parameters: Robust concepts and methods.

Han, X. and Chang, X. (2013). An intelligent noise reduction method for chaotic signals based on genetic algorithms and lifting wavelet transforms. *Information Sciences*, 218:103–118.

Harris, T. (2013). Quantitative credit risk assessment using support vector machines: Broad versus narrow default definitions. *Expert Systems with Applications*, 40(11):4404–4413.

Hastie, T. J. and Tibshirani, R. J. (2017). Generalized additive models.

Hennig, C. (1999). Models and methods for clusterwise linear regression. In *Classification in the Information Age*, pages 179–187. Springer.

Horn, D., Demircioğlu, A., Bischl, B., Glasmachers, T., and Weihs, C. (2018). A comparative study on large scale kernelized support vector machines. *Advances in Data Analysis and Classification*, 12(4):867–883.

Ikeda, K. and Murata, N. (2005). Effects of norms on learning properties of support vector machines. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 5, pages v–241. IEEE.

Kalcsics, J., Nickel, S., Puerto, J., and Tamir, A. (2002). Algorithmic results for ordered median problems. *Operations Research Letters*, 30(3):149–158.

Labbé, M., Martínez-Merino, L. I., and Rodríguez-Chía, A. M. (2019). Mixed integer linear programming for feature selection in support vector machine. *Discrete Applied Mathematics*, 261:276–304.

Labbé, M., Ponce, D., and Puerto, J. (2017). A comparative study of formulations and solution methods for the discrete ordered p-median problem. *Computers & Operations Research*, 78:230–242.

Lauer, F. and Guermeur, Y. (2011). Msvmpack: a multi-class support vector machine package. *The Journal of Machine Learning Research*, 12:2293–2296.

Lee, Y., Lin, Y., and Wahba, G. (2004). Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81.

Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Lichman, M. et al. (2013). Uci machine learning repository.

López, J., Maldonado, S., and Carrasco, M. (2018). Double regularization methods for robust feature selection and svm classification via dc programming. *Information Sciences*, 429:377–389.

Majid, A., Ali, S., Iqbal, M., and Kausar, N. (2014). Prediction of human breast and colon cancers from imbalanced data using nearest neighbor and support vector machines. *Computer methods and programs in biomedicine*, 113(3):792–808.

Maldonado, S., Bravo, C., López, J., and Pérez, J. (2017). Integrated framework for profit-based feature selection and svm classification in credit scoring. *Decision Support Systems*, 104:113–121.

Mangasarian, O. (1999). Arbitrary-norm separating plane. *Operations Research Letters*, 24(1-2):15–23.

Marín, A., Martínez-Merino, L. I., Puerto, J., and Rodríguez-Chía, A. M. (2021). The soft-margin support vector machine with ordered weighted average. *Knowledge-Based Systems*, page 107705.

Martini, H. and Schöbel, A. (1998). Median hyperplanes in normed spaces—a survey. *Discrete Applied Mathematics*, 89(1-3):181–195.

Martini, H. and Schöbel, A. (2001). Median and center hyperplanes in minkowski spaces—a unified approach. *Discrete Mathematics*, 241(1-3):407–426.

McGee, V. E. and Carleton, W. T. (1970). Piecewise regression. *Journal of the American Statistical Association*, 65(331):1109–1124.

Megiddo, N. and Tamir, A. (1983). Finding least-distances lines. *SIAM Journal on Algebraic Discrete Methods*, 4(2):207–211.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C., Lin, C.-C., and Meyer, M. D. (2019). Package 'e1071'. *The R Journal*.

Nalepa, J. and Kawulok, M. (2019). Selecting training sets for support vector machines: a review. *Artificial Intelligence Review*, 52(2):857–900.

Natarajan, N., Dhillon, I. S., Ravikumar, P., and Tewari, A. (2017). Cost-sensitive learning with noisy labels. *J. Mach. Learn. Res.*, 18(1):5666–5698.

Nickel, S., Puerto, J., Rodríguez-Chía, A., and Weissler, A. (2005). Multicriteria planar ordered median problems. *Journal of Optimization Theory and Applications*, 126(3):657–683.

Northcutt, C., Jiang, L., and Chuang, I. (2021). Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411.

Ogryczak, W. and Tamir, A. (2003). Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122.

Park, Y. W., Jiang, Y., Klabjan, D., and Williams, L. (2017). Algorithms for generalized clusterwise linear regression. *INFORMS Journal on Computing*, 29(2):301–317.

Platt, J. C., Cristianini, N., Shawe-Taylor, J., et al. (1999). Large margin dags for multiclass classification. In *nips*, volume 12, pages 547–553.

Quandt, R. E. (1958). The estimation of the parameters of a linear regression system obeying two separate regimes. *Journal of the american statistical association*, 53(284):873–880.

Quinlan, J. (1996). Machine learning and id3. *Morgan Kauffman, Los Altos*.

Radhimeenakshi, S. (2016). Classification and prediction of heart disease risk using data mining techniques of support vector machine and artificial neural network. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 3107–3111. IEEE.

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880.

Ryan, D. M. and Foster, A. (1981). An integer programming approach to scheduling. In Wren, A., eds., *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. North-Holland, Amsterdan.

Salzberg, S. L. (1994). C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993.

Schöbel, A. (1996). Locating least-distant lines with block norms. *Studies in Locational Analysis*, 10:139–150.

Schöbel, A. (1997). Locating line segments with vertical distances. *Studies in Locational Analysis*, 11:143–158.

Schöbel, A. (1998). Locating least-distant lines in the plane. *European Journal of Operational Research*, 106(1):152–159.

Schöbel, A. (2003). Anchored hyperplane location problems. *Discrete and Computational Geometry*, 29(2):229–238.

Schöbel, A. (2013). *Locating lines and hyperplanes: theory and algorithms*, volume 25. Springer Science & Business Media.

Schöbel, A. (2015). Location of dimensional facilities in a continuous space. In *Location Science*, pages 135–175. Springer.

Späth, H. (1982). A fast algorithm for clusterwise linear regression. *Computing*, 29(2):175–181.

Tukey, J. W. (1962). The future of data analysis. *The annals of mathematical statistics*, 33(1):1–67.

Üney, F. and Türkay, M. (2006). A mixed-integer programming approach to multi-class data classification problem. *European journal of operational research*, 173(3):910–920.

van den Burg, G. and Groenen, P. (2016). Gensvm: A generalized multiclass support vector machine. *Journal of Machine Learning Research*, 17:1–42.

Weber, A. (1909). Ueber den standort der industrien. erster teil. reine theorie der standorte. *Mohr, Tübingen*.

Weerasinghe, S., Erfani, S. M., Alpcan, T., and Leckie, C. (2019). Support vector machines resilient against training data integrity attacks. *Pattern Recognition*, 96:106985.

Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Technical report, Citeseer.

Yang, J., Wang, H., Wang, W., and Yu, P. (2003). Enhanced biclustering on expression data. In *Third IEEE Symposium on Bioinformatics and Bioengineering, 2003. Proceedings.*, pages 321–327. IEEE.