Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

On the Relationship among the MDM, SMO and SVM-Light Algorithms for Training Support Vector Machines

Master's thesis presented to apply for the Master
in Computer Engineering and Telecommunications degree

By

Jorge López Lázaro

under the direction of

José Ramón Dorronsoro Ibero

Madrid, May 27, 2008

Contents

Abstract

Support Vector Machine training implies solving a quadratic programming problem, which is a computationally quite expensive task. For this reason, not only general nonlinear optimization techniques are employed, but also algorithms specially designed for SVMs. Among these, we centre our attention in two different approaches for solving the SVM duals: decomposition and geometrical methods. Although they have been traditionally considered as different approaches, we show theoretically and experimentally that there is a strong connection among two of the most widely used decomposition methods (SMO and SVM-Light) and a purely geometrical algorithm (MDM). This connection implies that it is possible to combine reasonings from both perspectives in order to gain deeper insights in the search of a more efficient algorithm for training SVMs.

Chapter 1

Introduction

Support Vector Machines (SVMs) were introduced in [1] and nowadays constitute one of the most prominent paradigms used for solving pattern recognition problems. Among the advantages they offer compared with other approaches, the following are worth mentioning: 1) unique (global) solution, 2) good generalization properties based on structural risk minimization, 3) a common ground for both classification/regression and linearly separable/non-linearly separable tasks, 4) a common ground as well to obtain both linear/nonlinear solutions.

Concentrating our attention on classification tasks, the non-linearly separable case is addressed by introducing slack variables in the primal problem, whereas they are not present in the linearly separable one. These two cases are known in the literature as soft-margin and hard-margin classification, respectively [2]. The aim is to find the best separating hyperplane between the two classes of training samples, where ”best” is to be understood as the one with a maximal margin, that is, the one further from the samples (patterns). In the non-separable case, however, there will not be any properly separating hyperplane, so slack variables are introduced to allow for classification errors. The best hyperplane is then considered as the one that provides a best balance between distance from the patterns to it and subsequent classification errors using this plane as a decision boundary.

As most real-life problems will not be linearly separable, the most straightforward option is to make always use of the soft-margin formulation. Nevertheless, there is a more powerful possibility known as the ”kernel trick”. Simply speaking, this option consists of mapping the patterns to a high dimensional feature space (with more dimensions as the original one, perhaps even infinite) hoping that they will be linearly separable after the mapping. Then we can solve the problem as if it were a hard-margin one obtaining thus in the original space not a hyperplane, but a hypersurface. This is possible since the original space formulation of SVMs only involves scalar products among the patterns; it can be shown that if we substitute those scalar products for a kernel function that satisfies Mercer's conditions [3] we will be implicitly mapping the patterns in that way.

1.1 Motivation

Despite the advantages mentioned in the previous section, SVMs have a remarkable drawback in the fact that they imply solving a large constrained quadratic programming (QP) problem, its size making often standard solvers impractical as memory requirements are quadratic in the number of patterns. This has given rise to a great deal of SVM specific training algorithms, which nearly always do not solve the original SVM problem (known as the primal), but an equivalent formulation in the dual space (dual problem), where the optimal Lagrangian multipliers are found (cf. §2). As a result, the solution hyperplane can be expressed as a linear combination of the patterns associated with non-zero multipliers (support vectors).

The big picture is that there are two main classes of training algorithms (see §3):

- Decomposition methods. The idea here is not to solve the whole dual problem, but iteratively select a subset of the Lagrangian multipliers and optimize over that subset fixing temporarily the values of the unselected multipliers.
- Geometry-based ones, which exploit the geometrical interpretation of SVMs: hard-margin classification (and also soft-margin squared penalty) can be viewed as computing the closest points in the convex hulls of the patterns from both classes, whereas soft-margin linear penalty classification corresponds to doing the same on reduced convex hulls.

Unfortunately, these two approaches to SVM training have been traditionally considered separately, only related in the final aim they pursue, which is solving the QP problem. The main motivation for this work is an observation we came up with while revising the state-of-the-art literature for both trends. Specifically, we realised that a geometrical method called MDM (presented in [4] and applied to SVM training in [5]) had a very similar scheme to the one of the SMO [6] algorithm, which has been, and still is, one of the most popular decomposition methods. This is a very interesting fact, since it implies that decomposition and geometrical methods sometimes share common ideas. Therefore, some decomposition methods can hopefully be improved with geometrical reasonings, and vice versa.

1.2 Objectives

The basic objectives of this work are thus the following:

- Implement the MDM algorithm following the lines of [4] and [5].
- Characterize theoretically the relationship between MDM and SMO.
- Confirm experimentally this relationship.
- Explore whether this relationship can be extended to other decomposition and/or geometrical methods.

Therefore, the work is mostly theoretical and the experiments are performed just in order to confirm or refute the formal derivations. It will be seen throughout the different chapters that these four objectives have been covered (see the structure in §1.4). The fourth one has been covered with another decomposition method called SVM-Light (see §3).

1.3 Contributions

The theoretical (cf. §4) and experimental (cf. §5) results we have achieved prove the following:

- If MDM and SMO choose the same patterns for updating, their updates are identical.
- The heuristics originally used in SMO by Platt are unnecessarily complex for what they are implicitly looking for, that is, finding the couple of patterns that violate the most the optimality conditions.
- MDM is designed to look directly for that couple of patterns (in the hull formulation).
- If the heuristics of SMO are simplified to look directly for that couple, and SMO is adapted to solve the problem covered by MDM (this implies that both patterns must belong to the same class), then both methods become the same algorithm.
- Finding the most violating couple gives the direction of steepest descent (among the directions expressed by two patterns), if we take a first-order approximation of the dual function. Thus, both algorithms can be considered to choose their update directions with a feasible direction strategy.
- SMO is a particular case of SVM-Light, since this last method uses the a feasible direction method to find the direction of steepest descent.

Some of these contributions have already been covered in the literature. For instance, in [7] a modification of SMO is proposed to look for the most violating couple (this is the one we use for our derivations). Besides, we have been given feedback -after the time of this writing- that the relationship between SMO and SVM-Light had been established in [8], but also that their relationship with MDM apparently has not been established so far. Therefore, the work has a novel component indeed.

1.4 Structure

The structure of this work consists of six chapters, whose organization and topics are as follows:

- Chapter 1: Introduction. The current chapter; it presents a general introduction to support vector machines, as well as the motivation, objectives and contributions of the work.

- Chapter 2: Support Vector Classification. Here the different formulations for support vector machines are presented: hard and soft-margin, linear and nonlinear, and primal and dual. The geometrical reformulation of the duals is also discussed.

- Chapter 3: State-of-the-art in Support Vector Machine Training. This chapter explains the distinction between decomposition and geometrical methods. In the decomposition group, the SMO and SVM-Light algorithms are analyzed, and the same is done for the MDM algorithm in the geometrical one.

- Chapter 4: CH-MDM and its Relationship with Decomposition Methods. Here is this work's theoretical core. First, an adaptation of MDM for the problem with two hulls is presented. Afterwards, this algorithm is interpreted as a decomposition method and put into relationship with SMO. Finally, it is interpreted as a feasible direction method and put into relationship with SVM-Light.

- Chapter 5: Experimental Results. Related to the previous chapter, here the derivations in it are experimentally shown to be true. Basically, the relationships SMO-MDM and SMO-SVM-Light are checked. Besides, it presents a comparison of soft-margin SVMs with linear and quadratic penalties.

- Chapter 6: Discussion and Additional Work. In this final chapter, some consequences of the theoretical and experimental results obtained in chapters 4 and 5 are discussed. Besides, the chapter ends with an enumeration of the different articles and papers that have been written throughout the development of this work, as well as some pointers to ideas that will be considered for future research.

Chapter 2

Support Vector Classification

2.1 Hard-margin Case

The primal and dual QP problems for the hard-margin SVM are respectively as follows [1]:

$$\begin{aligned} &\min_{\mathbf{w},b}\ \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &s.t.\quad y_i\left(\mathbf{w}\cdot\mathbf{x_i}+b\right)\geq 1\ \ \forall i, \end{aligned} \tag{2.1.1}$$

$$\begin{aligned} &\min_{\alpha}\ W\left(\alpha\right)=-\textstyle\sum_{i=1}^{N}\alpha_i+\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\mathbf{x_i}\cdot\mathbf{x_j} \\ &s.t.\quad \alpha_i\geq 0\ \ \forall i,\ \ \textstyle\sum_{i=1}^{N}\alpha_i y_i=0. \end{aligned} \tag{2.1.2}$$

The KKT conditions can be summarized in the following way [9]:

$$\mathbf{w}^*=\textstyle\sum_{i\in SV}\alpha_i^* y_i\mathbf{x_i}, \tag{2.1.3}$$

$$y_i\left(\mathbf{w}^*\cdot\mathbf{x_i}+b^*\right)=1\ \ \forall i\in SV, \tag{2.1.4}$$

$$y_i\left(\mathbf{w}^*\cdot\mathbf{x_i}+b^*\right)\geq 1\ \ \forall i\in NSV, \tag{2.1.5}$$

where SV denotes the set of indices belonging to support vectors, i.e. patterns with $\alpha_i^*>0$, and NSV the analogous set for non-support vectors, i.e. patterns with $\alpha_i^*=0$.

Because of optimality theory, the fulfilment of the KKT conditions and of the dual constraints is a necessary and sufficient condition for the point to be the dual global minimum [9]. This is valid for the remaining SVM formulations in the present chapter.

2.2 Kernel Algorithms for Support Vector Machines

Observe that in (2.1.2) the only operations between patterns are inner products. Besides, because of the KKT condition (2.1.3) the solution of (2.1.1) will be a classifier $(\mathbf{w}^*, b^*)$ that will classify a pattern $\mathbf{x_j}$ by computing the quantity $\mathbf{w}^* \cdot \mathbf{x_j} + b^* = \sum_{i \in SV} \alpha_i^* y_i \mathbf{x_i} \cdot \mathbf{x_j} + b^*$, which again only involves inner products between patterns.

At this point a useful result in [3] comes at hand, known as Mercer's theorem. Basically, it states that if a certain kernel function $k(\mathbf{x_i}, \mathbf{x_j})$ fulfils some properties such as positive definiteness, then $k(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$, where ϕ is a mapping from the input space to a higher dimensional one (perhaps even infinite) called the feature space.

In this way, by making use of a kernel function in (2.1.1) or (2.1.2) each time we need to calculate an inner product, we are effectively finding a hard-margin SVM in the feature space, which will become a non-linear decision boundary in the input space, being able thus to deal with inseparable classes in a very straightforward way.

Besides, if a certain algorithm used to solve either (2.1.1) or (2.1.2) is compelled to use inner products as the only operation between patterns, it is clear that it would be implicitly working in the feature space, yielding the aforementioned SVM.

In order to do this, two of the most popular kernel functions are the Gaussian and the polynomial ones, given respectively by the following formulæ:

$$k(\mathbf{x_i}, \mathbf{x_j}) = e^{-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}}, \tag{2.2.1}$$

$$k(\mathbf{x_i}, \mathbf{x_j}) = (1 + \mathbf{x_i} \cdot \mathbf{x_j})^p. \tag{2.2.2}$$

2.3 Soft-margin with Linear Penalties Case

However, there is no guarantee that by making use of a kernel function (2.1.1) will be feasible if the classes are not linearly separable. In what is known as the soft-margin 1-SVM, slack variables ξ_i are introduced in order to allow for classification errors, so now the classes need not be linearly separable as in the hard-margin case. The primal and dual QP problems analogous to (2.1.1) and (2.1.2) are:

$$\begin{aligned} &\min_{\mathbf{w}, b, \xi} \ \tfrac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\ &s.t. \quad y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 - \xi_i \ \ \forall i, \ \ \xi_i \geq 0. \ \forall i \end{aligned} \tag{2.3.1}$$

$$\begin{aligned} \min_\alpha \; W(\alpha) &= -\textstyle\sum_{i=1}^N \alpha_i + \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \\ s.t. \quad & 0 \leq \alpha_i \leq C \;\; \forall i, \;\; \textstyle\sum_{i=1}^N \alpha_i y_i = 0. \end{aligned} \tag{2.3.2}$$

Comparing (2.3.2) and (2.1.2) we see that the only difference is that now the multipliers are upper bounded by the parameter C. Having a look at the primal function, C determines the compromise between maximizing the margin and misclassifying patterns (those with $\xi_i \geq 1$) or correctly classifying the ones closer to the hyperplane than its margin (those with $0 < \xi_i \leq 1$).

The KKT conditions for this case give:

$$\mathbf{w}^* = \textstyle\sum_{i \in SV} \alpha_i^* y_i \mathbf{x_i}, \tag{2.3.3}$$

$$y_i \left(\mathbf{w}^* \cdot \mathbf{x_i} + b^*\right) = 1 \;\; \forall i \in SVNUB, \tag{2.3.4}$$

$$y_i \left(\mathbf{w}^* \cdot \mathbf{x_i} + b^*\right) \leq 1 \;\; \forall i \in SVUB, \tag{2.3.5}$$

$$y_i \left(\mathbf{w}^* \cdot \mathbf{x_i} + b^*\right) \geq 1 \;\; \forall i \in NSV, \tag{2.3.6}$$

where we have two new sets SVUB and SVNUB formed by the indexes belonging to support vectors whose coefficient respectively is or is not the upper bound, that is, C. Note that the inner products can be calculated again via a kernel function if we want to work in a feature space, and that now the feasibility of (2.3.1) is guaranteed due to the presence of the slack variables.

2.4 Soft-margin with Square Penalties Case

As for the soft-margin 2-SVM, this time the slack variables ξ_i are penalized quadratically, having thus the following primal and dual QP problems:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \; & \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{C}{2}\textstyle\sum_{i=1}^N \xi_i^2 \\ s.t. \quad & y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) \geq 1 - \xi_i \;\; \forall i, \end{aligned} \tag{2.4.1}$$

$$\begin{aligned} \min_\alpha \; W(\alpha) &= -\textstyle\sum_{i=1}^N \alpha_i + \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} + \frac{1}{2C}\sum_{i=1}^N \alpha_i^2 \\ s.t. \quad & 0 \leq \alpha_i \;\; \forall i, \;\; \textstyle\sum_{i=1}^N \alpha_i y_i = 0. \end{aligned} \tag{2.4.2}$$

The KKT conditions for this case give:

$$\mathbf{w}^* = \textstyle\sum_{i \in SV} \alpha_i^* y_i \mathbf{x_i}, \tag{2.4.3}$$

$$y_i \left(\mathbf{w}^* \cdot \mathbf{x_i} + b^*\right) = 1 - \tfrac{\alpha_i^*}{C} \;\; \forall i \in SV, \tag{2.4.4}$$

$$y_i \left(\mathbf{w}^* \cdot \mathbf{x_i} + b^*\right) \geq 1 \;\; \forall i \in NSV. \tag{2.4.5}$$

It is interesting to note that (2.4.2) is equivalent to (2.1.2), if we substitute the original kernel function -in order to cover the term $\frac{1}{2C}\sum_{i=1}^{N}\alpha_i^2$- by the new function $\tilde{k}\left(\mathbf{x_i}, \mathbf{x_j}\right) = k\left(\mathbf{x_i}, \mathbf{x_j}\right) + \frac{\delta_{ij}}{C}$, where δ_{ij} is the Kronecker delta symbol. This is the same result obtained in [10], who realized that by making the substitutions $\tilde{\mathbf{w}} = \begin{pmatrix} \mathbf{w} \\ \sqrt{C}\xi \end{pmatrix}$, $\tilde{b} = b$, $\tilde{\mathbf{x}}_\mathbf{i} = \begin{pmatrix} \mathbf{x_i} \\ \frac{y_i}{\sqrt{C}}\mathbf{e_i} \end{pmatrix}$ (2.4.1) becomes (2.1.1), where $\mathbf{e_i}$ denotes the N-dimensional vector in which the $i-th$ component is one and all the others are zero. Therefore, this case can be solved by an algorithm designed for the hard-margin case, with the additional advantage that (2.4.1) is always feasible due to the slack variables, regardless of the linear (in)separability of the patterns in the input space (or in the feature space if we use kernels).

Henceforth hard-margin and soft-margin with square penalties will be considered as equivalent problems.

2.5 Geometry of Support Vector Machines

As we will see in §3.1, in order to solve the previous QP problems there exist several decomposition algorithms that perform well. Nevertheless, some problems still arise, such as how to choose the parameter C in the soft-margin formulations, as it is a penalty term not very intuitively tuneable. Besides, these QP problems are not really "visual" in the sense that a non-expert user can hardly grasp what the algorithm is doing. For these and other reasons, [11, 12, 5] provide geometric reformulations of SVMs that allow us to reinterpret very intuitively the dual formulations. Besides, fast geometric algorithms not originally designed for SVM training can be adapted to solve them, as will be seen in §3.2.

2.5.1 Hard-margin and Soft-margin with Square Penalties as a Convex Hull Nearest Point Problem

If we make the substitutions $\mathbf{w} = \frac{2}{\|\tilde{\mathbf{w}}\|^2}\tilde{\mathbf{w}}$, $\beta = \frac{(-b+1)\|\tilde{\mathbf{w}}\|^2}{2}$, $\gamma = \frac{(-b-1)\|\tilde{\mathbf{w}}\|^2}{2}$ in (2.1.1) we obtain the following primal and dual QP problems:

$$\begin{gathered} \min_{\tilde{\mathbf{w}}, b} \; \tfrac{1}{2}\|\tilde{\mathbf{w}}\|^2 - \beta + \gamma \\ s.t. \;\; \tilde{\mathbf{w}} \cdot \mathbf{x_i} \geq \beta \;\; \forall i \in I_1, \;\; \tilde{\mathbf{w}} \cdot \mathbf{x_i} \leq \gamma \;\; \forall i \in I_2, \end{gathered} \tag{2.5.1}$$

$$\begin{aligned} &\min_{\tilde{\alpha}}\ W(\tilde{\alpha}) = \tfrac{1}{2}\textstyle\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{\alpha}_i\tilde{\alpha}_j y_i y_j \mathbf{x_i}\cdot\mathbf{x_j} \\ &s.t.\ \ \tilde{\alpha}_i \geq 0\ \ \forall i,\ \ \textstyle\sum_{i=1}^{N}\tilde{\alpha}_i y_i = 0,\ \ \sum_{i=1}^{N}\tilde{\alpha}_i = 2, \end{aligned} \tag{2.5.2}$$

where I_i denotes the set of indices of patterns that belong to class C_i. Making the previous substitutions in the KKT conditions (2.1.3)–(2.1.5) yields:

$$\tilde{\mathbf{w}}^* = \textstyle\sum_{i\in SV}\tilde{\alpha}_i^* y_i \mathbf{x_i}, \tag{2.5.3}$$

$$\tilde{\mathbf{w}}^*\cdot\mathbf{x_i} = \beta^*\ \ \forall i\in I_1\cap SV, \tag{2.5.4}$$

$$\tilde{\mathbf{w}}^*\cdot\mathbf{x_i} = \gamma^*\ \ \forall i\in I_2\cap SV, \tag{2.5.5}$$

$$\tilde{\mathbf{w}}^*\cdot\mathbf{x_i} \geq \beta^*\ \ \forall i\in I_1\cap NSV, \tag{2.5.6}$$

$$\tilde{\mathbf{w}}^*\cdot\mathbf{x_i} \leq \gamma^*\ \ \forall i\in I_2\cap NSV, \tag{2.5.7}$$

which are the KKT conditions for (2.5.1) and thus (2.1.1) and (2.5.1) are equivalent problems [11, 5]. Note that (2.5.2) can be seen as a convex hull nearest point problem (CH-NPP) where the objective is to find the closest points in the convex hulls of both classes, since we are minimizing the norm $\|\sum_{i=1}^{N}\tilde{\alpha}_i y_i \mathbf{x_i}\|^2$. Therefore, the orientation of the hyperplane is given by the optimal vector $\tilde{\mathbf{w}}^* = \tilde{\mathbf{w}}_\mathbf{1}^* - \tilde{\mathbf{w}}_\mathbf{2}^*$, where $\tilde{\mathbf{w}}_\mathbf{1}^* = \sum_{i\in I_1}\tilde{\alpha}_i^*\mathbf{x_i}$, $\sum_{i\in I_1}\tilde{\alpha}_i^* = 1$, $\tilde{\alpha}_i^* \geq 0$ and $\tilde{\mathbf{w}}_\mathbf{2}^* = \sum_{i\in I_2}\tilde{\alpha}_i^*\mathbf{x_i}$, $\sum_{i\in I_2}\tilde{\alpha}_i^* = 1, \tilde{\alpha}_i^* \geq 0$ give respectively the optimal points in $CH(C_1)$ and $CH(C_2)$. As for its position, it is made to bisect the segment between $\tilde{\mathbf{w}}_\mathbf{1}^*$ and $\tilde{\mathbf{w}}_\mathbf{2}^*$ and thus it is easy to infer by means of a simple geometric argument that the bias must be $\tilde{b}^* = \frac{\|\tilde{\mathbf{w}}_\mathbf{2}^*\|^2 - \|\tilde{\mathbf{w}}_\mathbf{1}^*\|^2}{\|\tilde{\mathbf{w}}^*\|^2}$ [5].

2.5.2 Soft-margin with Linear Penalties as a Reduced Convex Hull Nearest Point Problem

It also turns out in [11] that (2.3.1) can be recast in geometrical terms. Using the same substitutions specified in §2.5.1, together with $\mu = \frac{C\|\tilde{\mathbf{w}}\|^2}{2}$, $\tilde{\xi}_i = \frac{\xi_i\|\tilde{\mathbf{w}}\|^2}{2}$, in (2.3.1) yields the new primal and dual QP problems:

$$\begin{aligned} &\min_{\tilde{\mathbf{w}},b,\xi}\ \tfrac{1}{2}\|\tilde{\mathbf{w}}\|^2 - \beta + \gamma + \mu\textstyle\sum_{i=1}^{N}\tilde{\xi}_i \\ &s.t.\ \ \tilde{\mathbf{w}}\cdot\mathbf{x_i} \geq \beta - \tilde{\xi}_i\ \ \forall i\in I_1,\ \ \tilde{\mathbf{w}}\cdot\mathbf{x_i} \leq \gamma + \tilde{\xi}_i\ \ \forall i\in I_2,\ \ \tilde{\xi}_i \geq 0\ \ \forall i, \end{aligned} \tag{2.5.8}$$

$$\begin{aligned} &\min_{\tilde{\alpha}}\ W(\tilde{\alpha}) = \tfrac{1}{2}\textstyle\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{\alpha}_i\tilde{\alpha}_j y_i y_j \mathbf{x_i}\cdot\mathbf{x_j} \\ &s.t.\ \ 0 \leq \tilde{\alpha}_i \leq \mu\ \ \forall i,\ \ \textstyle\sum_{i=1}^{N}\tilde{\alpha}_i y_i = 0,\ \ \sum_{i=1}^{N}\tilde{\alpha}_i = 2. \end{aligned} \tag{2.5.9}$$

Making the previous substitutions in the KKT conditions (2.3.3)–(2.3.6) this time yields:

$$\tilde{\mathbf{w}}^* = \textstyle\sum_{i \in SV} \tilde{\alpha}_i^* y_i \mathbf{x_i}, \tag{2.5.10}$$

$$\tilde{\mathbf{w}}^* \cdot \mathbf{x_i} = \beta^* \;\; \forall i \in I_1 \cap SVNUB, \tag{2.5.11}$$

$$\tilde{\mathbf{w}}^* \cdot \mathbf{x_i} = \gamma^* \;\; \forall i \in I_2 \cap SVNUB, \tag{2.5.12}$$

$$\tilde{\mathbf{w}}^* \cdot \mathbf{x_i} \leq \beta^* \;\; \forall i \in I_1 \cap SVUB, \tag{2.5.13}$$

$$\tilde{\mathbf{w}}^* \cdot \mathbf{x_i} \geq \gamma^* \;\; \forall i \in I_2 \cap SVUB, \tag{2.5.14}$$

$$\tilde{\mathbf{w}}^* \cdot \mathbf{x_i} \geq \beta^* \;\; \forall i \in I_1 \cap NSV, \tag{2.5.15}$$

$$\tilde{\mathbf{w}}^* \cdot \mathbf{x_i} \leq \gamma^* \;\; \forall i \in I_2 \cap NSV, \tag{2.5.16}$$

which are the KKT conditions for (2.5.8) and thus (2.3.1) and (2.5.8) are equivalent problems [11, 12]. Note that (2.5.9) can be seen as a reduced convex hull nearest point problem (RCH-NPP) by the same reasoning as before, since now $\tilde{\mathbf{w}}_{\mathbf{1}}^* = \sum_{i \in I_1} \tilde{\alpha}_i^* \mathbf{x_i},\ \sum_{i \in I_1} \tilde{\alpha}_i^* = 1,\ 0 \leq \tilde{\alpha}_i^* \leq \mu$ and $\tilde{\mathbf{w}}_{\mathbf{2}}^* = \sum_{i \in I_2} \tilde{\alpha}_i^* \mathbf{x_i},\ \sum_{i \in I_2} \tilde{\alpha}_i^* = 1,\ 0 \leq \tilde{\alpha}_i^* \leq \mu$ give respectively the optimal points in $RCH\,(C_1)$ and $RCH\,(C_2)$.

2.5.3 Support Vector Classification as a Minimal Norm Problem

In the work [5] there is an additional contribution regarding the geometry of SVMs. Once we have the duals (2.5.2) and (2.5.9), we can build the Minkowski difference of the (reduced) convex hulls, that is, the set $Z = \{\mathbf{w_1} - \mathbf{w_2},\ \mathbf{w_1} \in (R)CH\,(C_1)\,,\ \mathbf{w_2} \in (R)CH\,(C_2)\}$. It is clear that if we try to minimize $\|z\|^2,\ z \in Z$ we have a minimal norm problem (MNP) equivalent to (2.5.2) (to (2.5.9)).

This point of view has the advantages that the solution is unique in terms of the coefficients α_i^* -something which is not necessarily true in a (R)CH-NPP, where $\mathbf{w}^*$ is unique, but maybe different values for α_i^* give it-, and that a new kind of algorithms can be used to train SVMs (see §3.2.1). Its major drawback is that MNP looks simpler than (R)CH-NPP, but actually the set Z is a convex polytope that can have up to $N_1 N_2$ vertices, where $N_i = |I_i|$, so this can become a very hard problem if the set Z needs explicitly be built.

Chapter 3

State-of-the-art in Support Vector Machine Training

3.1 Decomposition Algorithms

As we mentioned in chapter §1, decomposition methods for training SVMs rely on the idea of not solving the whole dual QP problem, but on solving iteratively (with a specific QP solver or another SVM algorithm) for a subset of multipliers. Therefore, the original QP problem is broken down into a series of smaller QP subproblems. If these subproblems are small enough, no space problems will arise even if the original QP to be solved is very large in terms of the number of patterns.

However, as a consequence of the decomposition, each iteration will not produce much progress towards the optimum, so the multipliers chosen should be those that allow us to progress as much as possible towards the solution of the global QP. This set of chosen multipliers is popularly known as the working set. Each algorithm belonging to this category uses a different strategy to determine which multipliers are to be included in the working set. From these algorithms, the best-known are chunking [13], a preliminary decomposition method by Osuna et al. [14], SMO [6] and SVM-Light [15]. In chunking, the size of the working set tends to grow with time, though it can occasionally shrink. In the method by Osuna, the size of the working set is fixed in advance and constant with time. As for SMO and SVM-Light, the following sections explain both methods in more detail.

3.1.1 SMO Algorithm

This method is intended to solve the soft-margin with linear penalties dual formulation (2.3.2) (recall that it can also be applied to solve (2.1.2) or (2.4.2) by taking $C = \infty$). It takes the strategy introduced by Osuna to its extreme by fixing the size of the working set to 2. This is the smallest possible size for it, since otherwise the constraint $\sum_{i=1}^{N} \alpha_i y_i = 0$ may not be fulfilled. It turns out that by doing this we can find the solution analytically, so that an additional solver is not necessary. This is the main advantage of SMO, as each iteration needs very little time once the two multipliers of the current working set have been chosen.

In order to choose them, Platt [6] proposed two heuristics. As a result of these heuristics, one of the multipliers is always associated with a pattern that violates some of the KKT conditions (2.3.4)–(2.3.6), for this ensures that each iteration reduces the overall dual function (2.3.2). A parameter $\epsilon > 0$ is introduced for relaxing, so that these conditions become:

$$1 - \epsilon \leq y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) \leq 1 + \epsilon \;\; \forall i \in SVNUB, \tag{3.1.1}$$

$$y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) \leq 1 + \epsilon \;\; \forall i \in SVUB, \tag{3.1.2}$$

$$y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) \geq 1 - \epsilon \;\; \forall i \in NSV. \tag{3.1.3}$$

The following definitions for the decision function and the classification error are also introduced:

$$f\left(\mathbf{x_i}\right) = \mathbf{w} \cdot \mathbf{x_i} + b = \textstyle\sum_{j \in SV} \alpha_j y_j \mathbf{x_j} \cdot \mathbf{x_i} + b, \tag{3.1.4}$$

$$E_i = f\left(\mathbf{x_i}\right) - y_i. \tag{3.1.5}$$

To be precise, Platt's heuristics operate as follows:

1. The first heuristic examines by default not the entire training set, but only the patterns in SVNUB, determining whether each of these violates or not the relaxed KKT conditions (3.1.1)–(3.1.3). If a violating pattern is found, it is immediately eligible for optimization. The second heuristic is launched to find the second multiplier. If the second heuristic is not able to find a second multiplier, the first heuristic chooses the next violating pattern in SVNUB. In the case that the second heuristic cannot find a second multiplier for any violating pattern in SVNUB (or that no pattern in SVNUB violates the relaxed KKT conditions), the first heuristic repeats the same process but over the entire training set, not only over SVNUB. In the extreme case where the second heuristic cannot either find a second multiplier for any choice of the first heuristic (or where there is no violating first pattern), SMO terminates since the progress in the dual function is considered to be negligible. If it can indeed find

a second multiplier, next time the first heuristic will examine again just the patterns in SVNUB.

2. The second heuristic tries to maximize the step taken during the joint optimization of the multipliers chosen by both heuristics. Assuming that the first heuristic has chosen α_i and that the second one is currently examining α_j, the step size is approximated by computing E_i, E_j and choosing an α_k such that $|E_i - E_k| = \max_j \{|E_i - E_j|\}$. If as a result of this choice, not enough progress is obtained after the optimization (see [6] for the details concerning what is considered to be enough progress), the second heuristic iteratively chooses a pattern in SVNUB. If again none of these is able to provide enough progress, the second heuristic iterates over the whole training set. If no training pattern is good enough, the first heuristic is resumed to look for another α_i, as explained above.

Refer to section §4.2.1 for what the operations performed by SMO during a joint optimization are, as well as for a discussion about these heuristics and another ones proposed more recently in [7].

3.1.2 SVM-Light Algorithm

This method is also intended to solve the soft-margin with linear penalties dual formulation (2.3.2). It is more general than SMO and similar to the method by Osuna, so that the working set size is set in advance to a fixed number, say q, with the constraint that this number is even. The main contribution of SVM-Light, apart from its numerous computational tricks that make it very fast, is the way to select the multipliers for the working set, by solving the following problem:

$$\begin{gathered}\min_{\mathbf{d}} \nabla_\alpha W(\alpha) \cdot \mathbf{d} \\ s.t. \quad \mathbf{y} \cdot \mathbf{d} = 0, \\ d_i \geq 0 \ \ \forall i \in NSV, \\ d_i \leq 0 \ \ \forall i \in SVUB, \\ -1 \leq d_i \leq 1 \ \ \forall i, \\ |\{d_i | d_i \neq 0\}| = q. \end{gathered} \tag{3.1.6}$$

Here $\mathbf{d}$ represents the steepest direction of descent determined by Zoutendijk's method [16] applied to a first-order approximation of the dual function. If a component d_i is not zero, it means that α_i is selected for the working set, otherwise that multiplier is frozen in the current iteration and does not take part in the QP subproblem. This approach seems to be more convenient than heuristics like the ones in SMO, but the additional optimization problem (3.1.6) has to be solved in each iteration of SVM-Light. Fortunately, the constraint that q is even makes this problem automatically solvable (cf. sections §4.3 and §4.3.1).

In order to make it faster, SVM-Light also has an own heuristic in order to decide which multipliers are likely to be associated to patterns in NSV or SVUB. If it were possible to know this in advance, all those patterns would never take part in the QP subproblems, as their multipliers' optimal values would have been correctly fixed. So we could train just over the patterns in SVNUB, making the training task much easier and faster.

This heuristic, known as shrinking, keeps a history of two Lagrange multipliers for every pattern. These Lagrange multipliers are the ones used to derive the KKT conditions $0 \leq \alpha_i$ and $\alpha_i \leq C$ by applying optimality theory [9]. The continuous positiveness of these multipliers is an estimate of the likelihood that a pattern belongs to NSV or SVUB, respectively. The patterns for which this happens are assumed to belong to NSV or SVUB, thus they do not take part in any QP subproblem from that moment. Since shrinking is a heuristic, it can fail, so SVM-Light's stopping criterion is based on the one in SMO, that is, checking the relaxed KKT conditions (3.1.1)–(3.1.3). If the conditions are violated by some of the "shrunk" patterns, the optimization continues with the violating patterns "unshrunk", i.e. being able to take part in the QP subproblems (see the details in [15]).

3.2 Geometry-based Algorithms

On the other hand, geometry-based methods exploit the geometric reinterpretation of SVM classifiers described in §2.5. These methods are often referred in the literature with the generic name of nearest point algorithms (NPA). These are iterative algorithms that asymptotically approximate either the closest point of a convex hull to the origin, either the closest points belonging to two different convex hulls, or the closest points belonging to two different reduced convex hulls, depending on whether the MNP, CH-NPP or RCH-NPP SVM reformulation is exploited.

The most common research procedure in this field consists of coming up with a MNP solver, adapting it to cover the CH-NPP problem, and finally trying to generalize this adaptation to cover also the RCH-NPP case. This procedure has given rise to the following triplet of algorithms: Gilbert's algorithm (MNP solver) [17], Schlesinger-Kozinec's -SK- algorithm (adaptation of Gilbert's one to CH-NPP) [18], and RCH-SK algorithm (adaptation of the latter to RCH-NPP, at first partially adapted in [19], and recently totally adapted in [20]).

A recent line of investigation consists of adapting Mitchell-Demyanov-Malozemov's -MDM- MNP solver [4], since it is remarkably faster than Gilbert's one. The following section explains this method in detail, paving the way for CH-MDM, which is studied in the next chapter.

3.2.1 Unary MDM Algorithm

As it was originally presented in [4], the aim of this algorithm is to solve iteratively a MNP in a convex polyhedron, that is, minimizing $\|\mathbf{w}\|^2$ for $\mathbf{w} \in CH(A)$, where $A = \{\mathbf{x_i}\}$, $i = 1, \cdots, N$. Observe that this is not the same problem described in section §2.5.3, where the convex polyhedron used is $Z = CH\left(\tilde{Z}\right)$, with $\tilde{Z} = \mathbf{x_i} - \mathbf{x_j}$, $i \in I_1$, $j \in I_2$, although if we managed somehow to work implicitly with the set $\tilde{Z}$ instead of A then we would be solving (2.5.2).

The algorithm calculates a sequence of approximations $\mathbf{w^{(t)}}$ to $\mathbf{w}^*$ in the following way: starting with an arbitrary $\mathbf{w^{(0)}} \in CH(A) \Rightarrow \mathbf{w^{(0)}} = \sum_{i=1}^{N} \alpha_i^{(0)} \mathbf{x_i}$, $\sum_{i=1}^{N} \alpha_i^{(0)} = 1$, an appropriate direction of movement $\mathbf{d^{(0)}}$ is searched, where:

$$\mathbf{d^{(t)}} = \mathbf{x_L^{(t)}} - \mathbf{x_U^{(t)}}, \tag{3.2.1}$$

$$\mathbf{x_L^{(t)}} = argmin_{\mathbf{x_i}}\left(\mathbf{w^{(t)}} \cdot \mathbf{x_i}\right), \tag{3.2.2}$$

$$\mathbf{x_U^{(t)}} = argmax_{\mathbf{x_i}|\alpha_i>0}\left(\mathbf{w^{(t)}} \cdot \mathbf{x_i}\right). \tag{3.2.3}$$

Putting this into relationship with the concept of margin, if we work with a zero bias term b the hyperplane is forced to pass through the origin and $\mathbf{x_L^{(0)}}$ is the pattern with least margin from the hyperplane that has as its normal vector $\mathbf{w^{(0)}}$, whereas $\mathbf{x_U^{(0)}}$ is the support vector with most margin. The intuitive idea why this direction is chosen is that $\mathbf{x_U^{(0)}}$ may not be a support vector in the optimum, as it is far from the current hyperplane.

Now the following approximation $\mathbf{w^{(1)}}$ is calculated as the point with least norm in the segment joining $\mathbf{w^{(0)}}$ and $\overline{\mathbf{w}}^{(0)}$, this last point being the one obtained as a result of removing $\mathbf{x_U}$ from the representation of $\mathbf{w^{(0)}}$ while moving along the direction $\mathbf{d^{(0)}}$. That is, $\overline{\mathbf{w}}^{(0)} = \mathbf{w^{(0)}} + \alpha_U^0 \mathbf{d^{(0)}}$. The same process is carried out for any iteration t, because $\mathbf{d^{(t)}}$ is guaranteed to be a descent direction. To see this, let $g(r)$ denote the norm of a point in the aforementioned segment:

$$g(r) = \|\left(1-r\right)\mathbf{w^{(t)}} + r\overline{\mathbf{w}}^{(t)}\|^2, \quad 0 \leq r \leq 1. \tag{3.2.4}$$

Then $g\prime(0) = 2\alpha_U^{(t)} \mathbf{w^{(t)}} \cdot \mathbf{d^{(t)}} \leq 0$, provoking thus a descent unless we are at the optimum. If we force $g\prime(r) = 0$, we get the value $r^* = \min\left\{1, \frac{\Delta^{(t)}}{\alpha_U^{(t)}\|\mathbf{d^{(t)}}\|^2}\right\}$, where:

$$\Delta^{(t)} = -\mathbf{w^{(t)}} \cdot \mathbf{d^{(t)}} = \mathbf{w^{(t)}} \cdot \mathbf{x_U^{(t)}} - \mathbf{w^{(t)}} \cdot \mathbf{x_L^{(t)}}. \tag{3.2.5}$$

We get thus the movement $\mathbf{w^{(t+1)}} = \mathbf{w^{(t)}} + \lambda \mathbf{d^{(t)}}$, where:

$$\lambda = r^* \alpha_U^{(t)} = \min\left(\alpha_U^{(t)}, \tfrac{\Delta^{(t)}}{\|\mathbf{d^{(t)}}\|^2}\right). \tag{3.2.6}$$

Since strictly speaking the algorithm is only working with the coefficients α_i, this movement obviously implies the following simple update:

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} \;\; \forall i \neq L, U, \tag{3.2.7}$$

$$\alpha_L^{(t+1)} = \alpha_L^{(t)} + \lambda, \tag{3.2.8}$$

$$\alpha_U^{(t+1)} = \alpha_U^{(t)} - \lambda. \tag{3.2.9}$$

There are several possible stopping criteria for this algorithm, but in [4] no particular criterion is proposed. One of the simplest consists of checking that $\Delta \approx 0$, since $\mathbf{w}^* \cdot \mathbf{x_U^*} = \mathbf{w}^* \cdot \mathbf{x_L^*}$. In the next chapter, more criteria are discussed for CH-MDM in section §4.1.3.

Chapter 4

CH-MDM and its Relationship with Decomposition Methods

As it has been seen in the previous chapter, Gilbert's algorithm gave rise to the CH-SK algorithm when the CH-NPP was directly considered instead of the MNP, and recently this has been also extended to cover the reduced convex hull case by the RCH-SK algorithm.

However, Gilbert's algorithm is much slower than MDM [5], the reason being that it is quite costly to get rid of wrong support vectors that have entered the active set, that is, to remove mistaken initial choices $\mathbf{x_i}$ by achieving $\alpha_i = 0$. The MDM algorithm can, on the other hand, easily dispose of wrong SV choices since in (3.2.9) α_U becomes zero when $\lambda = \alpha_U$. So it would be very convenient to generalize MDM in the same way, producing far more efficient solvers for CH-NPP and RCH-NPP than their counterparts CH-SK and RCH-SK.

This has already been carried out, but only up to a certain point. In [5] the MDM algorithm was adapted to solve CH-NPP (2.5.2) by working implicitly with the polyhedron $\tilde{Z}$. Nonetheless, working with the Minkowski difference of the convex hulls is quite less visual than working with both hulls separately. The adaptation that is proposed in section §4.1 for the CH-NPP is really straightforward and it is performed in a very similar way than the one for CH-SK.

Regarding the generalization for RCH-NPP, the only attempt that seems to have been tried so far is the very recent one proposed in [21], which makes use of the projection theorem stated in [19] and [20]. Nevertheless, this generalization is not fully satisfactory, because its movements inside the reduced convex hull are not the same as the ones performed by CH-MDM inside a standard hull (in this approach the point used as $\mathbf{x_U}$ is not a vertex of the RCH as it should be, but one of the CH), so the advantages that CH-MDM offers are partially lost. It can be seen that RCH-SK, on the contrary, performs equivalently to its counterpart CH-SK (this comes from the fact that CH-SK also uses $\mathbf{x_L}$, which is indeed a vertex of the RCH in RCH-SK). In short, coming up with a properly generalized RCH-MDM is still an open problem.

After obtaining CH-MDM, the algorithm is considered under analytic and feasible direction perspectives in sections §4.2 and §4.3, respectively. These observations help us to establish clear relationships between CH-MDM and the two decomposition methods described in the previous chapter: SMO and SVM-Light. To our knowledge, the consideration of CH-MDM from the point of view of feasible directions is a pioneer one. The attempt to put into relationship decomposition and geometric algorithms, instead of considering them as different approaches (which is what has been done so far in the literature that has been examined), seems to be also a novelty.

4.1 Implementation of CH-MDM

This section presents the pseudocode for an implementation of MDM for CH-NPP, following the lines of the one given in [18]. The first three subsections deal with some preliminary observations about how to calculate the margins, how to choose the update direction and how to stop. After them, the fourth subsection contains the proposed pseudocode.

4.1.1 Calculating the margins

Firstly, as we are now working with two convex hulls the projections have to be calculated along $y_i\mathbf{w}$, because if $\mathbf{x_i} \in C_1$ then the origin of the MNP problem becomes $\mathbf{w_2}$ and if $\mathbf{x_i} \in C_2$ it becomes $\mathbf{w_1}$. In this way, the projections for the first class are along the vector $\mathbf{w} = \mathbf{w_1} - \mathbf{w_2}$ and along $-\mathbf{w} = \mathbf{w_2} - \mathbf{w_1}$ for the second class. Besides, this change in the origin forces us to change the patterns for $\mathbf{x_i} - \mathbf{w_2}$ if $\mathbf{x_i} \in I_1$ and $\mathbf{x_i} - \mathbf{w_1}$ if $\mathbf{x_i} \in I_2$. In summary, the margin m for a pattern $\mathbf{x_i}$ is now calculated as follows:

$$m\left(\mathbf{x_i}\right) = \frac{(\mathbf{w_1}-\mathbf{w_2})\cdot(\mathbf{x_i}-\mathbf{w_2})}{\|\mathbf{w}\|} \quad \forall i \in I_1, \tag{4.1.1}$$

$$m\left(\mathbf{x_i}\right) = \frac{(\mathbf{w_2}-\mathbf{w_1})\cdot(\mathbf{x_i}-\mathbf{w_1})}{\|\mathbf{w}\|} \quad \forall i \in I_2. \tag{4.1.2}$$

4.1.2 Choosing the Update Direction

Secondly, as a consequence of the constraints $\sum_{i=1}^{N} \alpha_i y_i = 0$ and $\sum_{i=1}^{N} \alpha_i = 2$, the patterns $\mathbf{x_L}$ and $\mathbf{x_U}$ must belong to the same class (otherwise, apart from α_U and α_L, two additional coefficients would have to be changed so as not to get out of the hulls). This implies that they cannot be simply the patterns with minimal and maximal margins as in §3.2.1. If the minimal and maximal-margin patterns do not belong to the same class, three immediate possibilities come to mind:

1. Set $\mathbf{x_L} = argmin_{\mathbf{x_i}} \{m(\mathbf{x_i})\}$ and $\mathbf{x_U} = argmax_{\mathbf{x_i} \mid i \in SV \wedge i \in I(\mathbf{x_L})} \{m(\mathbf{x_i})\}$
2. Set $\mathbf{x_U} = argmax_{\mathbf{x_i} \mid i \in SV} \{m(\mathbf{x_i})\}$ and $\mathbf{x_L} = argmin_{\mathbf{x_i} \mid i \in I(\mathbf{x_U})} \{m(\mathbf{x_i})\}$
3. Set $\mathbf{x_{U_1}} = argmax_{\mathbf{x_i} \mid i \in SV \cap I_1} \{m(\mathbf{x_i})\}$, $\mathbf{x_{L_1}} = argmin_{\mathbf{x_i} \mid i \in I_1} \{m(\mathbf{x_i})\}$, as well as $\mathbf{x_{U_2}} = argmax_{\mathbf{x_i} \mid i \in SV \cap I_2} \{m(\mathbf{x_i})\}$, $\mathbf{x_{L_2}} = argmin_{\mathbf{x_i} \mid i \in I_2} \{m(\mathbf{x_i})\}$. Set $\Delta_1 = y_{U_1}\mathbf{w} \cdot \mathbf{x_{U_1}} - y_{L_1}\mathbf{w} \cdot \mathbf{x_{L_1}}$ and $\Delta_2 = y_{U_2}\mathbf{w} \cdot \mathbf{x_{U_2}} - y_{L_2}\mathbf{w} \cdot \mathbf{x_{L_2}}$ and take $\mathbf{x_U}$ and $\mathbf{x_L}$ from $\Delta = \max\{\Delta_1, \Delta_2\}$.

These choices clearly correspond to taking the class of $\mathbf{x_L}$, taking the class of $\mathbf{x_U}$ and taking the class with maximal Δ, respectively. From now on, we will use the last possibility. Whatever the choice, once $\mathbf{x_L}$ and $\mathbf{x_U}$ have been chosen, in order to be consistent with (3.2.5) Δ is clearly seen to be defined as:

$$\Delta = \|\mathbf{w}\| \left(m(\mathbf{x_U}) - m(\mathbf{x_L})\right). \tag{4.1.3}$$

4.1.3 Choosing When to Stop

Thirdly, it has not been said yet when MDM stops (it cannot be run indefinitely, since it only converges asymptotically to the optimal point). Several stopping criteria have been proposed in the literature about NPP algorithms, based on different geometrical reasonings after choosing an accuracy parameter ϵ:

1. Stop if $\Delta \leq \epsilon\|\mathbf{w}\|^2$, since at the optimum $\Delta^* = 0$ [4].
2. Stop if $\Delta \leq 2\epsilon\|\mathbf{w}\|^2$, since this implies that the KKT conditions are fulfilled with accuracy $\epsilon\|\mathbf{w}\|^2$ (see below).
3. Stop if $2\|\mathbf{w}\| - (\min_{i \in I_1}\{m(\mathbf{x_i})\} + \min_{i \in I_2}\{m(\mathbf{x_i})\}) \leq \epsilon\|\mathbf{w}\|$, since this difference is 0 at the optimum [5, 19].
4. Stop if $\|\mathbf{w}\| - \min_i\{m(\mathbf{x_i})\} \leq \epsilon\|\mathbf{w}\|$, since this difference is also 0 at the optimum [18, 20].

Note that some of these criteria (e.g. [18, 19]) were originally conceived comparing with ϵ and not with $\epsilon\|\mathbf{w}\|$, so they were based on computing an absolute error, whereas these are based on relative errors. We find this approach more consistent, for the norm of the optimum vector will depend on the problem being addressed and now it is this norm what is being approximated with relative accuracy ϵ [5].

Note as well that the first and the second criteria are equivalent, but the reasoning used to give them is quite different. The first one is based on the geometrical fact that $\mathbf{w}^* \cdot \mathbf{x_U} = \mathbf{w}^* \cdot \mathbf{x_L}$, whereas the second criterion is explained next.

Without loss of generality, let us see how $\Delta_1 \leq 2\epsilon$ must be satisfied to fulfil the KKT conditions with an accuracy ϵ (the reasoning is analogous for Δ_2). Consider the conditions (2.5.4) and (2.5.6) . In order to find an approximate solution they can be relaxed like this, by introducing an $\epsilon > 0$:

$$\beta - \epsilon \leq \mathbf{w} \cdot \mathbf{x_i} \leq \beta + \epsilon \;\; \forall i \in I_1 \cap SV, \tag{4.1.4}$$

$$\beta - \epsilon \leq \mathbf{w} \cdot \mathbf{x_i} \;\; \forall i \in I_1 \cap NSV. \tag{4.1.5}$$

Since $y_i = +1$ we have that $\mathbf{w} \cdot \mathbf{x_i} = y_i \mathbf{w} \cdot \mathbf{x_i} \;\; \forall i \in I_1$. From equation (4.1.1), and by introducing the term $\mathbf{w} \cdot \mathbf{w_2}$ in (4.1.4) and (4.1.5) we get:

$$\beta - \epsilon - \mathbf{w} \cdot \mathbf{w_2} \leq \|\mathbf{w}\| m\left(\mathbf{x_i}\right) \leq \beta + \epsilon - \mathbf{w} \cdot \mathbf{w_2} \;\; \forall i \in I_1 \cap SV,$$

$$\beta - \epsilon - \mathbf{w} \cdot \mathbf{w_2} \leq \|\mathbf{w}\| m\left(\mathbf{x_i}\right) \;\; \forall i \in I_1 \cap NSV.$$

According to the definitions for $\mathbf{x_{U_1}}$ and $\mathbf{x_{L_1}}$, it goes without saying that $\mathbf{x_{U_1}}$ is the pattern that violates the most the first equation's right inequality (if it does violate it). As for $\mathbf{x_{L_1}}$, it is the one that violates the most either the first equation's left inequality or the second equation's only inequality (if it does violate them). Observe that the KKT conditions provide us with an additional reasoning, aside from geometry, to choose $\mathbf{x_{U_1}}$ and $\mathbf{x_{L_1}}$ as we do; the former in $SV \cap I_1$ and the latter in I_1.

Now, if the bias is taken to bisect the segment between $\mathbf{w_1}$ and $\mathbf{w_2}$, i.e. $b = \frac{\|\mathbf{w_2}\|^2 - \|\mathbf{w_1}\|^2}{\|\mathbf{w}\|^2}$, it is easy to see from the definition of β in section §2.5.1 that $\beta = \mathbf{w} \cdot \mathbf{w_1}$. Therefore, if the above conditions are to be satisfied the following must be true:

$$\|\mathbf{w}\| m\left(\mathbf{x_{U_1}}\right) \leq \|\mathbf{w}\|^2 + \epsilon,$$

$$\|\mathbf{w}\|^2 - \epsilon \leq \|\mathbf{w}\| m\left(\mathbf{x_{L_1}}\right).$$

Using (4.1.3) implies that $\Delta_1 \leq 2\epsilon$, which is what we wanted to show. Replacing ϵ with $\epsilon\|\mathbf{w}\|^2$ in the reasoning makes the relaxation relative instead of absolute and yields $\Delta_1 \leq 2\epsilon\|\mathbf{w}\|^2$, which, together with $\Delta_2 \leq 2\epsilon\|\mathbf{w}\|^2$, justifies the second stopping criterion.

4.1.4 Final implementation

At this point we are ready to provide an implementation for the MDM algorithm. The lines of [18, 20] regarding notation and general scheme will be followed. The next variables will be used and can be cached so as to accelerate the execution:

$$
\begin{aligned}
A &= \mathbf{w_1} \cdot \mathbf{w_1} = \textstyle\sum_{i\in I_1} \sum_{j\in I_1} \alpha_i\alpha_j \mathbf{x_i}\cdot\mathbf{x_j},\\
B &= \mathbf{w_2} \cdot \mathbf{w_2} = \textstyle\sum_{i\in I_2} \sum_{j\in I_2} \alpha_i\alpha_j \mathbf{x_i}\cdot\mathbf{x_j},\\
C &= \mathbf{w_1} \cdot \mathbf{w_2} = \textstyle\sum_{i\in I_1} \sum_{j\in I_2} \alpha_i\alpha_j \mathbf{x_i}\cdot\mathbf{x_j},\\
D_i &= \mathbf{w_1} \cdot \mathbf{x_i} = \textstyle\sum_{j\in I_1} \alpha_j \mathbf{x_i}\cdot\mathbf{x_j} \;\; \forall i,\\
E_i &= \mathbf{w_2} \cdot \mathbf{x_i} = \textstyle\sum_{j\in I_2} \alpha_j \mathbf{x_i}\cdot\mathbf{x_j} \;\; \forall i.
\end{aligned}
$$

Note that the algorithm only involves inner products between patterns, so that it can be directly kernelized with the procedure explained in §2.2. The updating rules for A, B, C, D_i, E_i are easily obtained from (3.2.7), (3.2.8), (3.2.9) and the definitions of these variables. As for the value for λ in line 16, the reasoning is analogous to what was said to yield equation (3.2.6) (see also the following section), so that the algorithm ensures that $\mathbf{w_1} \in CH\left(C_1\right)$ and $\mathbf{w_2} \in CH\left(C_2\right)$.

4.2 CH-MDM as a Two Vector Decomposition Method

Recall from section §3.1 that for every iteration in a decomposition method there are some coefficients α_i which are "frozen", and that as a result the simpler optimization task over the non-frozen coefficients is solved. If we have a look at the updates in algorithm 1 we clearly see that, as in the unary case, the only coefficients to be updated are α_U and α_L, so the other ones are effectively frozen acting in the spirit of decomposition techniques.

To see that it is indeed a decomposition algorithm, it has to be shown that, once the non-frozen coefficients α_U and α_L have been chosen, the dual problem is optimized with respect to them. The problem in this case is (2.5.2), that is, minimizing $\|\mathbf{w}\|^2$, with $\mathbf{w} = \mathbf{w_1} - \mathbf{w_2} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}$. Without loss of generality, let us assume that $y_U = y_L = 1$. From what was said in section §3.2.1, after finding $\mathbf{x_L}$ and $\mathbf{x_U}$ the segment between $\mathbf{w_1}$ and $\overline{\mathbf{w}}_\mathbf{1} = \mathbf{w_1} + \alpha_U\left(\mathbf{x_L} - \mathbf{x_U}\right)$ is considered.

We can write analytically this segment in the form $\mathbf{w_1}\left(r\right) = \mathbf{w_1} + r\alpha_U\left(\mathbf{x_L} - \mathbf{x_U}\right),\ 0 \leq r \leq 1$. Since we want to optimize the norm over this segment while taking $\mathbf{w_2}$ as the origin, we can write:

Algorithm 1 CH-MDM algorithm for binary SVMs

```
1: Initialize every $\alpha_i$ so that $\mathbf{w_1} \in CH(C_1)$ and $\mathbf{w_2} \in CH(C_2)$.
2: Calculate $A, B, C$ as well as $D_i, E_i \ \forall i$.
3: loop
4:     Set $\mathbf{x_{U_1}} = argmax_{\mathbf{x_i} \mid i \in SV \cap I_1} \{m(\mathbf{x_i})\}$, $\mathbf{x_{L_1}} = argmin_{\mathbf{x_i} \mid i \in I_1} \{m(\mathbf{x_i})\}$.
5:     Set $\mathbf{x_{U_2}} = argmax_{\mathbf{x_i} \mid i \in SV \cap I_2} \{m(\mathbf{x_i})\}$, $\mathbf{x_{L_2}} = argmin_{\mathbf{x_i} \mid i \in I_2} \{m(\mathbf{x_i})\}$.
6:     Calculate $\Delta_1$ and $\Delta_2$.
7:     if $\Delta_1 \geq \Delta_2$ then
8:         Set $\mathbf{x_U} = \mathbf{x_{U_1}}, \mathbf{x_L} = \mathbf{x_{L_1}}, \Delta = \Delta_1$.
9:     else
10:        Set $\mathbf{x_U} = \mathbf{x_{U_2}}, \mathbf{x_L} = \mathbf{x_{L_2}}, \Delta = \Delta_2$.
11:    end if
12:    if stop criterion is true then
13:        Set $\mathbf{w} = \mathbf{w_1} - \mathbf{w_2}, \ b = \frac{\|\mathbf{w_2}\|^2 - \|\mathbf{w_1}\|^2}{\|\mathbf{w}\|^2}$.
14:        Return $\mathbf{w}, b$.
15:    end if
16:    Set $F = \|\mathbf{x_L} - \mathbf{x_U}\|^2, \lambda = \min\left\{\alpha_U, \frac{\Delta}{F}\right\}$.
17:    if $U, L \in I_1$ then
18:        $A \Leftarrow A + \lambda(2(D_L - D_U) + \lambda F)$
19:        $C \Leftarrow C + \lambda(E_L - E_U)$
20:        $D_i \Leftarrow D_i + \lambda(\mathbf{x_i} \cdot \mathbf{x_L} - \mathbf{x_i} \cdot \mathbf{x_U}) \ \forall i$
21:    else
22:        $B \Leftarrow B + \lambda(2(E_L - E_U) + \lambda F)$
23:        $C \Leftarrow C + \lambda(D_L - D_U)$
24:        $E_i \Leftarrow E_i + \lambda(\mathbf{x_i} \cdot \mathbf{x_L} - \mathbf{x_i} \cdot \mathbf{x_U}) \ \forall i$
25:    end if
26:    $\alpha_L \Leftarrow \alpha_L + \lambda$
27:    $\alpha_U \Leftarrow \alpha_U - \lambda$
28: end loop
```

$$\|\mathbf{w_1}(r) - \mathbf{w_2}\|^2 = \|\mathbf{w}\|^2 + (r\alpha_U\|\mathbf{x_L} - \mathbf{x_U}\|)^2 - 2r\alpha_U\mathbf{w} \cdot (\mathbf{x_U} - \mathbf{x_L}). \tag{4.2.1}$$

Differentiating and equalling to zero yields $r^* = \frac{\mathbf{w} \cdot (\mathbf{x_U} - \mathbf{x_L})}{\alpha_U\|\mathbf{x_U} - \mathbf{x_L}\|^2}$. Using (4.1.3) gives $r^* = \frac{\Delta}{\alpha_U\|\mathbf{x_U} - \mathbf{x_L}\|^2}$, $0 \leq r^* \leq 1$. Note that the first inequality is guaranteed since both the numerator and denominator are greater or equal than zero, but not the second one. Now if we define $\lambda = \alpha_U r^*$ and ensure that $r^* \leq 1$ we have the same expression used in line 16 of algorithm 1.

The reasoning for the case where $y_U = y_L = -1$ is analogous. Therefore, the dual is optimized with respect to α_U and α_L, which is what was to be shown.

4.2.1 Relationship between CH-MDM and SMO

Recall from section §3.1.1 that SMO takes the decomposition strategy to its extreme by choosing only two multipliers in each iteration. This choice of two multipliers makes it clear that this method has a strong relationship with CH-MDM, since we have just seen how CH-MDM also solves analytically the resulting task on two multipliers for a CH-NPP. In this section we characterize what this relationship exactly consists of.

SMO as it stands in [6] solves the dual of the soft-margin 1-SVM formulation, that is, (2.3.2). To show the relationship I will follow the lines of [6] and section 7.5 of [9]. First, we have to know the operations performed by SMO in a joint optimization of the two multipliers it chooses, say, for convenience, α_1 and α_2.

Because of the problem constraints $\sum_{i=1}^{N} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$, it is fairly clear that $\alpha_1^{new}, \alpha_2^{new} \in [U, V]$, where:

$$U = \max\left\{0, \alpha_1^{old} + \alpha_2^{old} - C\right\}, \quad V = \min\left\{C, \alpha_1^{old} + \alpha_2^{old}\right\} \quad if\ y_1 = y_2, \tag{4.2.2}$$

$$U = \max\left\{0, \alpha_2^{old} - \alpha_1^{old}\right\}, \quad V = \min\left\{C, C - \alpha_1^{old} + \alpha_2^{old}\right\} \quad if\ y_1 \neq y_2, \tag{4.2.3}$$

and $\alpha_i^{old}, \alpha_i^{new}$ denote respectively the current (i.e. before the optimization) and new (i.e. after it) values of the multiplier α_i. We can state the following theorem taken from [9] (refer to the proof in that book or in [6]), which specifies the optimization carried out by SMO:

Theorem 1 *Let* $\kappa = \|\mathbf{x_1} - \mathbf{x_2}\|^2$*, and* E_i *defined in equation* (3.1.5)*. Assuming without loss of generality that we optimize first over* α_2*, we get the optimal value* $\alpha_2^{new} = \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\kappa}$*. After cutting this* α_2^{new} *(if necessary) to be in the interval* $[U, V]$*, the optimal value for the other multiplier will be* $\alpha_1^{new} = \alpha_1^{old} + y_1 y_2 \left(\alpha_2^{old} - \alpha_2^{new}\right)$*.*

Recall that the problem (2.5.2) solved by CH-MDM is a rescaling of (2.3.2) in which in every moment $C = \infty$ and $\sum_{i=1}^{N} \alpha_i = 2$. Interestingly, despite of this implicit rescaling, the updates performed by both algorithms have the same form, as stated in the following novel result:

Theorem 2 *Assume that SMO and CH-MDM choose the same patterns while solving their respective problems, that is, $\alpha_2 = \alpha_L$, $\alpha_1 = \alpha_U$. Then, the updates carried out by both algorithms are identical.*

Proof Since $\alpha_1 = \alpha_U$ and $\alpha_2 = \alpha_L$, we have $y_1 = y_2$ because of CH-MDM's inherent constraint that $\mathbf{x_U}$ and $\mathbf{x_L}$ belong to the same class. This, together with $C = \infty$, produces the bounds $U = 0$ and $V = \alpha_1^{old} + \alpha_2^{old}$. As $y_1 y_2 = 1$ and using theorem 1, the update rule for SMO would be $\alpha_1^{new} = \alpha_1^{old} + \alpha_2^{old} - \alpha_2^{new}$ (after cutting α_2^{new} to lie in $[U, V]$).
As $E_1 = f(\mathbf{x_1}) - y_1$, $E_2 = f(\mathbf{x_2}) - y_2$ and $y_1 = y_2$, we get $E_1 - E_2 = f(\mathbf{x_1}) - f(\mathbf{x_2})$. Thus, applying that theorem:

$$\alpha_2^{new} = \alpha_2^{old} + \tfrac{y_2(f(\mathbf{x_1}) - f(\mathbf{x_2}))}{\kappa}.$$

Since $\alpha_1^{old} = \alpha_U$ and $\alpha_2^{old} = \alpha_L$ it is obvious from algorithm 1 that $\kappa = F = \|\mathbf{d}\|^2$. Using the facts that $f(\mathbf{x_i}) = \mathbf{w} \cdot \mathbf{x_i} + b$, together with $y_U = y_1 = y_2 = y_L$, makes $y_2 (f(\mathbf{x_1}) - f(\mathbf{x_2})) = y_U \mathbf{w} \cdot \mathbf{x_U} - y_L \mathbf{w} \cdot \mathbf{x_L} = \Delta$. Therefore, the rule for the uncut value is the same as $\alpha_2^{new} = \alpha_2^{old} + \frac{\Delta}{\|\mathbf{d}\|^2}$.
The last step is cutting this value. As $\frac{\Delta}{\|\mathbf{d}\|^2} \geq 0$ it is impossible that $\alpha_L^{new} < U = 0$. So the only cut is setting $\alpha_L^{new} = V$ if $\alpha_L^{old} + \frac{\Delta}{\|\mathbf{d}\|^2} > V = \alpha_L^{old} + \alpha_U^{old}$. This last condition implies that $\frac{\Delta}{\|\mathbf{d}\|^2} > \alpha_U^{old}$. Hence, the result of the cut is the same as the update rule

$$\alpha_L^{new} = \alpha_L^{old} + \min\left\{\alpha_U^{old}, \tfrac{\Delta}{\|\mathbf{d}\|^2}\right\}.$$

It is immediate that this rule and the one for $\alpha_U^{new} = \alpha_1^{new}$ are equivalent to the ones for MDM in (3.2.8), (3.2.9) and algorithm 1, proving thus the theorem.

This result is very intuitive if it is noted that both algorithms solve analytically for the two chosen multipliers, but from different points of view; MDM in a purely geometrical way and SMO without taking any geometrical consideration. At this point an interesting question to answer is whether SMO actually takes $\mathbf{x_U}$ and $\mathbf{x_L}$ for updating. The answer is that it does not necessarily choose them, at least from what the heuristics originally proposed by Platt do (cf. section §3.1.1).

Analyzing them we can see that, basically, the first heuristic takes any pattern that violates the relaxed KKT conditions (3.1.1)-(3.1.3); then the second one looks for a pattern that together with the other one causes a big enough descent in the dual. However, it will turn out in the following section that MDM chooses the best descent direction among all possible descent directions with two patterns. Moreover, as we have seen in §4.1.3 that it also takes the pair of points that violate the most (considered as a couple) the KKT conditions, the underlying complexity in the heuristics of SMO seems unnecessary for what is desired: MDM yields the biggest descent and at the same time updates the patterns which violate the most the KKT conditions, exactly the two goals pursued by the two heuristics, but in a much simpler and quicker way.

Similar ideas were used in [7] to propose two modifications of Platt's SMO. In the first modification, the first heuristic is not changed (say it chooses α_2 in a certain moment), whereas the second heuristic always chooses the multiplier α_1 associated to the pattern that maximizes the KKT violation produced by $\mathbf{x_1}$ and $\mathbf{x_2}$ as a couple. In the second modification, α_1 and α_2 correspond to the patterns that maximize as a couple the KKT violation. In both modifications other minor changes are present to check optimality (the stopping criterion is still checking the relaxed KKT conditions (3.1.1)-(3.1.3)) and to choose the bias.

These three SMO versions are increasingly simple: Platt's SMO tries to approximate the most violating couple in a very intricate way, Keerthi's modification 1 fixes the choice for α_1 to approximate the most violating couple once α_2 is chosen, and finally Keerthi's modification 2 fixes both α_1 and α_2 to the most violating couple, i.e. in theory what MDM does.

Besides, in [7] modification 2 is claimed to be the fastest method among the three versions. This fact is coherent with what was said above about MDM taking the biggest descent. The KKT conditions are not the same, because Keerthi's modification 2 solves for (2.3.2) and CH-MDM for (2.5.2), but both algorithms are expected to work very similarly if we take $C = \infty$ (more about this in the experimental section §5).

4.3 CH-MDM as an Improving Feasible Direction Method

To see that MDM is also an improving feasible direction technique, it is useful to revise some preliminary theory about feasible directions. I will follow the lines of [22], where we want to solve a linear-constraint optimization problem of the following general form:

$$\begin{aligned} &\min\nolimits_{\alpha}\ f(\alpha) \\ &s.t.\ \ \Theta\alpha \leq \theta,\ \ \Phi\alpha = \phi, \end{aligned} \tag{4.3.1}$$

where Θ and Φ are matrices for the inequality and equality constraints respectively, and θ and ϕ the corresponding vectors for the right terms of the constraints. Now it is fairly easy to reformulate the convex hull dual (2.5.2) in this matrix form, since the constraints are linear too:

$$\begin{aligned} &\min_\alpha \; W(\alpha) = \tfrac{1}{2}\alpha^T K \alpha \\ s.t. \quad & -I_N \alpha \leq \mathbf{0}, \quad \begin{pmatrix} \mathbf{y}^T \\ \mathbf{1}^T \end{pmatrix} \alpha = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \end{aligned} \tag{4.3.2}$$

where I_N stands for the N-dimensional identity matrix and $K_{ij} = y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}$. Next, we define the concepts of feasible and improving directions:

Definition 1 *Assume that α is a feasible point for problem* (4.3.1) *(i.e. it fulfils the problem constraints). A direction $\mathbf{d}$ is said to be feasible if $\exists \delta > 0$ such that $\alpha + r\mathbf{d}\ \forall r \in (0, \delta)$ is still a feasible point.*

Definition 2 *Assume that α is a point (not necessarily feasible) for problem* (4.3.1). *A direction $\mathbf{d}$ is said to be improving if $\exists \delta > 0$ such that $f(\alpha + r\mathbf{d}) < f(\alpha)\ \forall r \in (0, \delta)$.*

We are ready to reproduce for completeness the following lemma from [22] (without proof):

Lemma 1 *Assume that α is a feasible point for problem* (4.3.1) *such that the inequality constraint matrix Θ can be decomposed into Θ_1 and Θ_2 satisfying $\Theta_1 \alpha = \theta_1$ and $\Theta_2 \alpha < \theta_2$, where $\theta^T = \left(\theta_1^T \theta_2^T\right)$. Then, a direction $\mathbf{d} \neq \mathbf{0}$ is feasible if and only if $\Theta_1 \mathbf{d} \leq \mathbf{0}$ and $\Phi \mathbf{d} = \mathbf{0}$. Furthermore, it is improving if $\nabla_\alpha f(\alpha) \cdot \mathbf{d} < 0$.*

What this lemma is saying in our context is that, if we want to find a feasible direction, we need only care about the coefficients α_i for which the inequality constraint becomes an equality (i.e. those related to Θ_1 and θ_1) and about satisfying the condition $\Theta_1 \mathbf{d} \leq \mathbf{0}$. Besides, the condition $\Phi \mathbf{d} = \mathbf{0}$ means that the right term of each equality constraint becomes zero if we use $\mathbf{d}$ instead of α. Once both conditions are fulfilled, we only need that the inner product between the gradient of the objective function and the improving direction be less than zero in order this direction to be also an improving one.

Turning our attention to (4.3.2) it is clear that Θ_1 is formed by the rows i of $-I_N$ such that α_i is a non-support vector, because the inequality $\alpha_i \leq 0$ becomes $\alpha_i = 0$. Conversely, Θ_2 is formed by the rows corresponding to support vectors. Let us denote these submatrices of $-I_N$ as $-I_{NSV}$ and $-I_{SV}$ respectively, so that we seek $-I_{NSV}\mathbf{d} \leq \mathbf{0}$. The equality constraints become now $\sum_{i=1}^N y_i d_i = 0$ and $\sum_{i=1}^N d_i = 0$. In order to find the most improving direction we can think of minimizing $\nabla_\alpha f(\alpha) \cdot \mathbf{d}$, where for this case $\nabla_\alpha f(\alpha) = K\alpha$.

This idea is correct, but observe that once we find an improving feasible direction $\mathbf{d}$ the direction $\overline{\mathbf{d}} = r\mathbf{d},\ r > 0$ is also improving and feasible and $\alpha^{\mathbf{T}} K \overline{\mathbf{d}} < \alpha^{\mathbf{T}} K \mathbf{d}$, so that we would be endlessly minimizing this product. Then, an additional constraint must be placed to bound $\mathbf{d}$. For example, in Zoutendijk's method [16] this constraint is $-\mathbf{1} \leq \mathbf{d} \leq \mathbf{1}$ (see chapter 10 in [22] for other possibilities). From all this, looking for the most improving feasible direction in CH-NPP becomes:

$$\begin{aligned} &\min_{\mathbf{d}}\ \alpha^T K \mathbf{d} \\ s.t.\quad &\mathbf{y} \cdot \mathbf{d} = 0,\ \ \mathbf{1} \cdot \mathbf{d} = 0, \\ &-I_{NSV} \mathbf{d} \leq 0, \\ &-1 \leq d_i \leq 1\ \ \forall i. \end{aligned} \tag{4.3.3}$$

By merging the two last constraints, this is trivially equivalent to:

$$\begin{aligned} &\min_{\mathbf{d}}\ F = \textstyle\sum_{i=1}^{N} (K\alpha)_i\, d_i \\ s.t.\quad &\textstyle\sum_{i=1}^{N} y_i d_i = 0,\ \ \sum_{i=1}^{N} d_i = 0, \\ &0 \leq d_i \leq 1\ \ \forall i \in NSV, \\ &-1 \leq d_i \leq 1\ \ \forall i \in SV. \end{aligned} \tag{4.3.4}$$

For the last step of the derivation, let us use a lemma stated and proved in [23], and adapt its notation as follows for the problem at hand:

Lemma 2 *If we need to optimize a function of the form $\sum_{i=1}^{N} w_i d_i$ subject to the following constraints:*

$$\begin{aligned} &\textstyle\sum_{i=1}^{N} d_i = 0, \\ &-1 \leq d_i \leq 1\ \ \forall i, \\ &|\,\{d_i | d_i \neq 0\}\,| = q, \\ &q = 2r,\ r \in \left[1, \lfloor \tfrac{N}{2} \rfloor\right], \end{aligned}$$

the solution is obtained in the way that follows. First, order the w_i in decreasing order. Secondly, take $d_i = -1, i = 1 \cdots r$ and $d_i = +1, i = N - r + 1 \cdots N$. This $\mathbf{d}$ gives the function's minimum value.

This lemma, though seeming complicated, is very intuitive by paying attention to what it says. We want to minimize a weighted function of some values w_i, where the corresponding weights are d_i. Note that q is the number of non-zero weights and that it is even. What should we do then? It is quite evident that in order to minimize the function we should weigh the biggest values with negative weights, whereas we should weigh the smallest values with positive ones. This is precisely what the lemma says: order the values w_i in decreasing order, associate $d_i = -1$ to the first $\frac{q}{2}$ of these (i.e. the biggest) and $d_i = +1$ to the last $\frac{q}{2}$ (i.e. the smallest). It is less clear that the q non-zero weights should be ± 1, but this is also shown to be true in [23].

Now let us check problem (4.3.4): the function to be minimized is also a weighted sum, where $w_i = (K\alpha)_i$. Moreover, this time the vector $\mathbf{d}$ describes also the descent direction to be chosen. We know from previous considerations that CH-MDM only implies two patterns that must belong to the same class, so in the context of the lemma $q = 2$. Whichever two patterns we do choose, the constraints $\sum_{i=1}^{N} y_i d_i = 0$ and $\sum_{i=1}^{N} d_i = 0$ become thus equivalent, so the only difference between problem (4.3.4) and the one stated in the lemma with $q = 2$ is that the vector $\mathbf{d}$ is more constrained. Despite this difference, we can state the following result:

Theorem 3 *CH-MDM is a feasible direction method, since it chooses at each step the optimizing direction by applying Zoutendijk's method to the dual* (2.5.2)*. Furthermore, this direction causes the steepest possible descent as a result of taking just two non-zero components.*

Proof We have proved above that (4.3.4) must be solved so as to find the best descent direction for the dual (2.5.2). It is trivial to note that, in order to satisfy $\sum_{i=1}^{N} d_i = 0$, there must be a negative d_i and a positive d_j. Assume for a moment that $-\mathbf{1} \leq \mathbf{d} \leq +\mathbf{1} \;\; \forall i$. The above lemma for $q = 2$ then says: order the values to be weighed, weigh the smallest value with -1 and the biggest one with $+1$.
With that assumption (4.3.4) would be solved by choosing $d_i = -1$ and $d_j = +1$, where $\mathbf{x_i} = argmax_{\mathbf{x_k}} \left\{ (K\alpha)_k \right\}$ and $\mathbf{x_j} = argmin_{\mathbf{x_k}} \left\{ (K\alpha)_k \right\}$. This is also seen with an analytic argument; the function to be minimized is:

$$\begin{aligned} F &= (K\alpha)_i d_i + (K\alpha)_j d_j \\ &= d_i \left((K\alpha)_i - (K\alpha)_j \right) \\ &= d_j \left((K\alpha)_j - (K\alpha)_i \right). \end{aligned}$$

Thus $\frac{\partial F}{d_i} = \left((K\alpha)_i - (K\alpha)_j\right)$. Since we have said that $d_i < 0$, we need to maximize $\left((K\alpha)_i - (K\alpha)_j\right) > 0$ so that the slope indicated by the derivative is as descending as possible. Conversely, $d_j > 0$, so we need to minimize $\left((K\alpha)_j - (K\alpha)_i\right) < 0$. This is obviously done by taking $\mathbf{x_i}$ and $\mathbf{x_j}$ as stated in the previous paragraph.

Rewriting

$$(K\alpha)_k = \sum_{m=1}^{N} \alpha_m K_{mk} = \sum_{m \in SV} \alpha_m K_{mk} = y_k \sum_{m \in SV} \alpha_m y_m \mathbf{x_m} \cdot \mathbf{x_k} = y_k \mathbf{w} \cdot \mathbf{x_k},$$

gives $\mathbf{x_j} = argmin_k \left(y_k \mathbf{w} \cdot \mathbf{x_k}\right)$. Note that setting $d_j = +1$ does not violate any constraint in (4.3.4) since the upper bound in the direction component is always $+1$.
Now we have to decide about $\mathbf{x_i}$. Noting that this time in (4.3.4) only support vectors can carry a negative d_i, it is not possible that $\mathbf{x_i} = argmax_k \left(y_k \mathbf{w} \cdot \mathbf{x_k}\right)$, as the lemma states. Instead, we need to restrict that $\mathbf{x_i} \in SV$ so that the direction remains feasible, hence $\mathbf{x_i} = argmax_{\mathbf{x_k} \mid \alpha_k > 0} \left(y_k \mathbf{w} \cdot \mathbf{x_k}\right)$.
Thus, following the spirit of lemma 2 and taking into account the additional constraints not covered in it, yields $\mathbf{x_i}$ and $\mathbf{x_j}$ as defined above. Forcing both points to belong to the same class by using one of the possibilities in section §4.1.2 clearly yields the points $\mathbf{x_U}$ and $\mathbf{x_L}$ used in CH-MDM, respectively.
Therefore, the optimal direction to minimize F is $\mathbf{d}^* = \lambda^* \left(\mathbf{x_L} - \mathbf{x_U}\right)$. Finally, we see that $\lambda^* = 1$, because if we took $d_j = +\lambda$ and $d_i = -\lambda$ for some $0 < \lambda \leq 1$, we would get $F = \lambda \left((K\alpha)_j - (K\alpha)_i\right)$. As we are interested in minimizing this value and the difference is by definition constant and negative, we would like to maximize λ, yielding the upper bound for it $\lambda^* = 1$. This is the same direction that CH-MDM takes in algorithm 1.

4.3.1 Relationship between CH-MDM and SVM-Light

Recall from section §3.1.2 that SVM-Light, like SMO, solves the dual of the soft-margin 1-SVM formulation (2.3.2), so we have to take $C = \infty$. The following corollary to theorem 3 is immediate:

Corollary 1 *Assume that SVM-Light is set to work with $q = 2$ and no shrinking. If the two patterns chosen by SVM-Light belong to the same class, i.e. $y_1 = y_2$, then the descent directions used by SVM-Light and CH-MDM are identical.*

Proof The best coefficients are again assumed to be α_1 and α_2. The fact that $C = \infty$ makes SVUB $= \phi$ in problem (3.1.6), which is solved by SVM-Light to find its best direction. Besides, the fact that $y_1 = y_2$ causes that the problem's constraint $\mathbf{y} \cdot \mathbf{d} = 0$ becomes

$\sum_{i=1}^{N} d_i = 0$. Thus, (3.1.6) becomes (4.3.4). The previous proof establishes that the best direction then is the one used by CH-MDM.

Note that it is guaranteed in this case that $\mathbf{x_1} = \mathbf{x_U}$ and $\mathbf{x_2} = \mathbf{x_L}$ because SVM-Light applies lemma 2 to solve (3.1.6) in the same spirit that in the previous proof, that is, using it to know how $\mathbf{x_1}$ and $\mathbf{x_2}$ should be chosen, and then using the additional constraints to restrict that choice.

Besides, if we use as stopping criterion for CH-MDM the one based on the KKT conditions (section §4.1.3) we will be stopping with the same criterion than SVM-Light, but applied to the relaxed conditions (2.5.4)–(2.5.7) instead of (3.1.1)–(3.1.3) as a result of considering $C = \infty$ and the rescaling of (2.1.2) to (2.5.2).

Chapter 5

Experimental Results

In this chapter we show experimentally some of the results obtained in the previous chapter about CH-MDM. Specifically, our aim is to study the performances of CH-MDM, Keerthi's modification 2 for SMO (cf. §4.2.1) and SVM-Light (cf. §3.1.2). We claimed in §4 that the three algorithms should behave very similarly if both SMO and SVM-Light are made to solve the 2-SVM dual problem (2.4.2) by means of making $C = \infty$ in the 1-SVM formulation. Recall that CH-NPP (2.5.2) is a rescaling of (2.4.2), so after CH-MDM solves CH-NPP, its solution should be the same than the one obtained with SMO and SVM-Light solving (2.4.2), once the rescaling is applied.

However, making $C = \infty$ not only provokes that the 1-SVM dual becomes the 2-SVM dual, but also causes that both problems become the hard-margin dual (2.1.2), because slack variables are infinitely penalized, so no classification errors are allowed. Therefore, if for some dataset the patterns are not linearly separable even with the use of kernels, the performances can be really poor. Thus, for the sake of completeness and good performances, we would like to compare somehow the 1-SVM and 2-SVM solutions separately, while at the same time confirming that the three algorithms have a strong relationship.

For the reasons above we adopt the following strategy: firstly, we adapt Keerthi's modification 2 for SMO to solve directly for the 2-SVM dual (henceforth this adaptation is referred to as 2-SMO). This is simple enough, since basically what has to be done is to modify the kernel to be $k(\mathbf{x_i}, \mathbf{x_j}) + \frac{\delta_{ij}}{C}$, and to consider the hard-margin KKT conditions (2.1.4) and (2.1.5) instead of the 1-SVM ones (2.3.4)–(2.3.6) while selecting the multipliers α_1 and α_2. Once the algorithm is adapted, CH-MDM and SMO can be properly compared in §5.2. Secondly, we do not modify SVM-Light nor set it to work with $C = \infty$. In this way, the solutions for 1-SVM and 2-SVM can be compared in §5.3 in a best-model fashion. Moreover, by making use of Keerthi's modification of SMO in its original form (referred to as 1-SMO) we are able to show that SVM-Light (with $q = 2$ and no shrinking) and 1-SMO are indeed the same algorithm, without using CH-MDM to put them into relationship. We have seen this last fact to be already known in another work whose existence we were not aware of at the time of this writing [8].

5.1 Datasets and test methodology

Before analyzing the results, we describe here the datasets and the test methodology used. Gaussian kernel support vector machines are employed, in order to deal with non-separability. We use 10 datasets taken from [24]: *titanic*, *heart*, *diabetes*, *breast cancer*, *thyroid*, *flare-solar*, *splice*, *image*, *german* and *banana*. All these datasets, except for *banana*, were originally taken from the UCI repository [25] and have been widely used as benchmark problems in the literature. Besides, in [24] each of these datasets is randomly partitioned into 100 different training-test pairs (except for *image* and *splice*, for which only 20 partitions are available), a fact that is very useful for our comparison purposes. Moreover, values C and σ are given for the 1-SVM formulation that result in good performance in terms of accuracy. These parameters are the ones 1-SMO and SVM-Light work with in §5.3.

The algorithms are compared over three different measures: accuracy of the final model in terms of error percentage, number of support vectors the SVMs have, and number of iterations needed to construct them. These are implementation-independent measures that also assess the quality of the model obtained (an iteration is to be understood as an optimization over two multipliers α_1 and α_2). It will be seen that, depending on the comparison, the stopping criteria are chosen differently for practical reasons. Once the measures are obtained for every dataset partition, Wilcoxon paired rank-sum tests in native R [26] are performed to see if there are statistically significant (at a 10% level) similarities and/or differences among the three algorithms. The rest of the implementation is written in C; the code for SVM-Light is the one written by Joachims and publicly available in [27], whereas the two versions of SMO are implemented from scratch, following the lines of the CH-MDM algorithm.

5.2 2-SMO versus CH-MDM

In this section we compare the performance of 2-SMO and CH-MDM with the parameter values $C = 10$ and $2\sigma^2 = 50$. While these are not optimal parameters, they give reasonable test accuracies and, therefore, are adequate enough for the comparisons made. Concerning the stopping criteria to be used, it has to be noted that no criterion from the geometry-based ones listed in §4.1.3 can in principle be used, since 2-SMO is solving for the 2-SVM dual, which does not have a direct geometric interpretation. Besides, the vectors $\mathbf{w}^{(t)}$ are different in both algorithms. Hence, it is convenient to choose a valid criterion aside from geometry. A possibility is observing that both problems are minimizing a dual; thus we can stop when the relative change in the duals is less than a tolerance ϵ. Specifically, this means for 2-SMO when $W\left(\alpha^{(t)}\right) - W\left(\alpha^{(t+1)}\right) \leq \epsilon W\left(\alpha^{(t)}\right)$, where $W\left(\alpha\right)$ is defined as in (2.4.2), and for CH-MDM when $\|\mathbf{w}^{(t)}\|^2 - \|\mathbf{w}^{(t+1)}\|^2 \leq \epsilon \|\mathbf{w}^{(t)}\|^2$.

Considering the derivations in [7] and in §4.2, it can be shown that if we want this version of SMO to update the coefficients in a similar way than CH-MDM (cf. algorithm 1), we have to redefine the "margins", the selection of $\mathbf{x_1}$ and $\mathbf{x_2}$ and the definition of Δ in the following way (from now on $\mathbf{x_1}$ and $\mathbf{x_2}$ will be referred to as $\mathbf{x_U}$ and $\mathbf{x_L}$ for analogy with CH-MDM):

$$m\left(\mathbf{x_i}\right) = \mathbf{w} \cdot \mathbf{x_i} - y_i, \tag{5.2.1}$$

$$\mathbf{x_L} = argmin_{\mathbf{x_i} \mid i \in SV \cup (NSV \cap I_1)} \left\{ m\left(\mathbf{x_i}\right) \right\}, \tag{5.2.2}$$

$$\mathbf{x_U} = argmax_{\mathbf{x_i} \mid i \in SV \cup (NSV \cap I_2)} \left\{ m\left(\mathbf{x_i}\right) \right\}, \tag{5.2.3}$$

$$\Delta = m\left(\mathbf{x_U}\right) - m\left(\mathbf{x_L}\right). \tag{5.2.4}$$

We see from theorem 1 and the definitions above that the updates in 2-SMO are of the form $\alpha_L^{new} = \alpha_L^{old} + y_L \frac{\Delta}{\|\mathbf{x_L} - \mathbf{x_U}\|^2}$ and $\alpha_U^{new} = \alpha_U^{old} + y_U y_L \left(\alpha_L^{old} - \alpha_L^{new}\right)$, since $E_U - E_L = m\left(\mathbf{x_U}\right) - m\left(\mathbf{x_L}\right) = \Delta$. As 2-SMO is not restrained to have $y_U = y_L$, there are four possible update cases depending on their values, not only two as happened with CH-MDM.

There are two further differences between both algorithms: the initialization and the bias calculation. The initialization in SMO can be arbitrary as long as $\sum_i \alpha_i y_i = 0$; we decided to implement 2-SMO so that it starts with two index choices for $\alpha_1 = 1$ and $\alpha_2 = 1$ such that $y_1 \neq y_2$, in order to have two support vectors from different classes (note that these values could be used as well to initialize CH-MDM). On the other hand, the bias is calculated by solving for the KKT condition (2.1.4) and taking for robustness the mean for all support vectors, as in the original SMO [6].

All the discussions above yield the following pseudocode, where the notation is the same as in CH-MDM algorithm 1. Observe now that the cuts for λ depend on which case we are dealing with, since λ can be subtracted from just α_U, just α_L, both or neither of them. Note as well that, as it stands, this pseudocode deals with the hard-margin SVM. In order to cope with a 2-SVM, the only modification is to substitute, as in CH-MDM, the inner products $\mathbf{x_i} \cdot \mathbf{x_j}$ for $k\left(\mathbf{x_i}, \mathbf{x_j}\right) + \frac{\delta_{ij}}{C}$, applying what was stated in §2.4. The calculation of the bias would still be valid, for the hard KKT condition (2.1.4) becomes the 2-SVM KKT condition (2.4.4).

Algorithm 2 2-SMO algorithm for binary SVMs

```
1: Choose arbitrarily $L, U$ so that $y_L \neq y_U$ and set $\alpha_L = \alpha_U = 1, \alpha_i = 0 \; \forall i \neq L, U$.
2: Calculate $A, B, C$ as well as $D_i, E_i \;\; \forall i$.
3: loop
4:     Set $\mathbf{x_L} = argmin_{\mathbf{x_i} \mid i \in SV \cup (NSV \cap I_1)} \{m(\mathbf{x_i})\}$.
5:     Set $\mathbf{x_U} = argmax_{\mathbf{x_i} \mid i \in SV \cup (NSV \cap I_2)} \{m(\mathbf{x_i})\}$.
6:     Set $\Delta = m(\mathbf{x_U}) - m(\mathbf{x_L})$.
7:     if stop criterion is true then
8:         Return $\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \mathbf{x_i}$, $b = \frac{1}{|SV|} \sum_{i \in SV} (y_i - \mathbf{w} \cdot \mathbf{x_i})$.
9:     end if
10:    Set $F = \|\mathbf{x_L} - \mathbf{x_U}\|^2, \lambda = \frac{\Delta}{F}$.
11:    if $U, L \in I_1$ then
12:        $\lambda \Leftarrow \min\{\lambda, \alpha_U\}$
13:        $A \Leftarrow A + \lambda(2(D_L - D_U) + \lambda F)$, $C \Leftarrow C + \lambda(E_L - E_U)$
14:        $D_i \Leftarrow D_i + \lambda(\mathbf{x_i} \cdot \mathbf{x_L} - \mathbf{x_i} \cdot \mathbf{x_U}) \;\; \forall i$
15:        $\alpha_L \Leftarrow \alpha_L + \lambda$, $\alpha_U \Leftarrow \alpha_U - \lambda$
16:    else if $U, L \in I_2$ then
17:        $\lambda \Leftarrow \min\{\lambda, \alpha_L\}$
18:        $B \Leftarrow B + \lambda(2(E_U - E_L) + \lambda F)$, $C \Leftarrow C + \lambda(D_U - D_L)$
19:        $E_i \Leftarrow E_i + \lambda(\mathbf{x_i} \cdot \mathbf{x_U} - \mathbf{x_i} \cdot \mathbf{x_L}) \;\; \forall i$
20:        $\alpha_L \Leftarrow \alpha_L - \lambda$, $\alpha_U \Leftarrow \alpha_U + \lambda$
21:    else if $U \in I_2, L \in I_1$ then
22:        $A \Leftarrow A + \lambda(2D_L + \lambda \mathbf{x_L} \cdot \mathbf{x_L})$, $B \Leftarrow B + \lambda(2E_U + \lambda \mathbf{x_U} \cdot \mathbf{x_U})$, $C \Leftarrow C +$
           $\lambda(E_L + D_U + \lambda \mathbf{x_L} \cdot \mathbf{x_U})$
23:        $D_i \Leftarrow D_i + \lambda \mathbf{x_i} \cdot \mathbf{x_L} \;\; \forall i$, $E_i \Leftarrow E_i + \lambda \mathbf{x_i} \cdot \mathbf{x_U} \;\; \forall i$
24:        $\alpha_L \Leftarrow \alpha_L + \lambda$, $\alpha_U \Leftarrow \alpha_U + \lambda$
25:    else
26:        $\lambda \Leftarrow \min\{\lambda, \alpha_L, \alpha_U\}$
27:        $A \Leftarrow A + \lambda(-2D_U + \lambda \mathbf{x_U} \cdot \mathbf{x_U})$, $B \Leftarrow B + \lambda(-2E_L + \lambda \mathbf{x_L} \cdot \mathbf{x_L})$, $C \Leftarrow C +$
           $\lambda(-E_U - D_L + \lambda \mathbf{x_L} \cdot \mathbf{x_U})$
28:        $D_i \Leftarrow D_i - \lambda \mathbf{x_i} \cdot \mathbf{x_U} \;\; \forall i$, $E_i \Leftarrow E_i - \lambda \mathbf{x_i} \cdot \mathbf{x_L} \;\; \forall i$
29:        $\alpha_L \Leftarrow \alpha_L - \lambda$, $\alpha_U \Leftarrow \alpha_U - \lambda$
30:    end if
31: end loop
```

Two different tolerances are used: $\epsilon = 0.001$ and $\epsilon = 0.00001$, whose respective results are given in tables 5.2.1 and 5.2.2. Let us first analyze table 5.2.1. In its results two facts are immediately noticeable and true for every dataset: CH-MDM gives models with significantly less support vectors than 2-SMO (except for *banana*), whereas 2-SMO requires significantly less iterations than its counterpart. This comes as no surprise: 2-SMO logically requires less iterations since the search for the two indices is not restrained by the condition $y_U = y_L$: therefore, the progress that 2-SMO makes in its dual is as large as the one in CH-MDM if that condition is fulfilled, and it is larger if the two patterns do not belong to the same class, because it is guaranteed that $\mathbf{x_U}$ and $\mathbf{x_L}$ constitute the KKT most violating couple.

However, CH-MDM arrives to SVM models with less number of support vectors. This somewhat makes up for the slower speed of this algorithm, as these are implicitly simpler models, whose predictions are faster; for every prediction there is one kernel operation per support vector, see (3.1.4). This fact is also easy to explain: CH-MDM is designed to get rid of unnecessary support vectors, by taking $\lambda = \alpha_U$ in line 16 of algorithm 1, and consequently updating with $\alpha_U = 0$. On the other hand, 2-SMO chooses λ depending of the case in algorithm 2; it is more difficult that this choice gets rid of a support vector, not only because this time there are three possible values for λ, but also because there are four different possible updates as a result of the freedom in choosing α_L and α_U. Eliminating mistaken support vectors in 2-SMO is a subproduct of the method, its main objective being optimizing its dual as much as possible.

Finally, the accuracy is generally better for 2-SMO; it beats CH-MDM in 6 datasets, whereas in the remaining 4 datasets both methods are statistically similar. Both algorithms are expected to yield the same solution if run indefinitely, as a consequence of the SVM solution uniqueness, so these divergences are conjectured to happen due to the tolerance level not being accurate enough. Since 2-SMO converges faster, it is logical to think that at this slightly inaccurate precision it gives better models than CH-MDM.

At this point it is useful to ascertain the validity of this surmise by having a look at table 5.2.2, where the tolerance level is 100 times more accurate. As it can be seen, the error percentages become much more similar: there are still significant differences in 5 of the datasets (4 for 2-SMO and 1 for CH-MDM), yet the difference in the means is not greater than 0.2%. This, together with the fact that the number of support vectors has also become almost equal in both methods, makes it clear that they are converging to the same optimal model. 2-SMO continues to converge significantly faster for all the datasets because of the aforementioned freedom in coefficient choices, so it seems to be a better algorithm than CH-MDM (at least for the chosen values for C and σ): if the precision is

Dataset	ERR. MDM	ERR. 2-SMO	SVs MDM	SVs 2-SMO	IT. MDM	IT. 2-SMO
Titanic	22.8±0.9	22.7±0.7[1]	121.5±12.7[1]	129.1±12.9	151.5±17.1	113.5±14.6[1]
Heart	19.5±3.2	19.1±3.1[1]	92.7±7.1[1]	97.7±7.1	133.8±14.9	117.0±9.7[1]
Diabetes	25.8±2.5	25.1±2.2[1]	290.2±14.7[1]	315.4±11.1	343.6±24.5	283.1±21.2[1]
Cancer	27.9±4.9	26.4±4.6[1]	152.1±6.7[1]	160.1±6.8	204.2±10.9	172.2±11.5[1]
Thyroid	7.3±2.8	7.1±2.6	64.2±8.4[1]	67.6±7.6	76.2±10.2	59.7±7.2[1]
Flare	38.9±5.5	37.6±5.7[1]	385.2±74.2[1]	420.1±80.8	405.5±76.3	267.9±56.8[1]
Splice	11.0±0.7	11.0±0.6	436.8±8.8[1]	488.6±11.8	435.6±9.1	266.0±6.8[1]
Image	8.7±1.8	9.3±3.1	352.8±28.6[1]	387.5±70.4	378.1±36.8	259.3±59.1[1]
German	26.4±2.6	25.2±2.2[1]	383.2±14.4[1]	446.3±16.8	436.9±24.8	368.5±27.0[1]
Banana	40.9±6.3	40.8±5.9	386.6±11.7	387.1±13.6	391.3±13.4	236.8±15.4[1]

Table 5.2.1: Average test accuracies, number of support vectors and number of iterations given by the CH-MDM and 2-SMO algorithms, with $\epsilon = 0.001$. A [1] stands for a statistically significant difference in the paired samples.

high it converges faster and gives equivalent models than those of CH-MDM, whereas if it is low it converges very quickly and the accuracy of the models makes up for their somewhat bigger complexity.

Dataset	ERR. MDM	ERR. 2-SMO	SVs MDM	SVs 2-SMO	IT. MDM	IT. 2-SMO
Titanic	22.6±0.7	22.6±0.7	149.0±2.4	149.1±2.3	292.2±18.6	238.2±14.9[1]
Heart	19.0±3.1	18.9±3.1[1]	105.7±8.5[1]	106.1±8.2	308.1±33.5	278.3±27.1[1]
Diabetes	24.0±1.9	23.9±1.8[1]	373.6±9.3[1]	374.3±9.9	940.8±67.7	817.4±66.3[1]
Cancer	27.1±4.8	26.9±4.8[1]	174.9±6.4[1]	175.4±6.4	485.1±42.8	432.6±31.4[1]
Thyroid	7.2±2.5	7.2±2.5	96.7±10.3[1]	97.2±10.1	148.3±14.9	130.8±12.1[1]
Flare	34.3±1.7	34.5±1.8	576.6±6.4	573.4±10.6[1]	1264.3±83.1	858.5±187.8[1]
Splice	10.7±0.7[1]	10.8±0.7	724.7±12.8	711.7±32.2[1]	957.0±35.2	655.2±89.9[1]
Image	6.0±0.8	6.0±0.8	576.3±43.7	577.5±27.1	1253.3±190.9	907.7±178.4[1]
German	24.6±2.2	24.5±2.2[1]	530.8±15.0[1]	535.7±13.4	1417.3±129.6	1244.8±107.2[1]
Banana	34.8±4.0	34.7±3.5	399.6±0.8	399.6±0.9	751.5±18.5	624.6±18.3[1]

Table 5.2.2: Average test accuracies, number of support vectors and number of iterations given by the CH-MDM and 2-SMO algorithms, with $\epsilon = 0.00001$. A [1] stands for a statistically significant difference in the paired samples.

5.3 1-SVM versus 2-SVM

As it has been said before, here we seek to compare the performance of 1-SVM and 2-SVM models over the different datasets we work with. 1-SVM models are trained both with SVM-Light (using the code in [27] without any change) and 1-SMO, whereas 2-SVM are trained both with CH-MDM and 2-SMO. Although this sort of comparison does not enlighten experimentally the relationship between SVM-Light and 1-SMO with CH-MDM (in order to do such a thing, we would have either to modify the code of SVM-Light to cover the 2-SVM case –something that in theory only implies changing the kernel function and setting $C = \infty$, but that is more complicated at code level– or to implement CH-MDM for 1-SVM –it would become RCH-MDM; recall that coming up with a proper RCH-MDM is still an open problem–), it is useful under three points of view: it allows us to keep on comparing CH-MDM and 2-SMO with different values for C and σ from those in section §5.2, it is as well designed to show the connection between 1-SMO and SVM-Light *per se* and, most importantly, to compare the performances of both SVM paradigms in a best-model fashion, i.e., with the best choice of parameters for each paradigm (or at least a very convenient one in terms of final accuracy).

Concentrating on the last point, there has been controversy in the literature about which paradigm is better: 1-SVM or 2-SVM. For instance, an advantage of 2-SVM that we have already seen is that it can be transformed to a hard-margin SVM, so every algorithm designed for solving CH-NPP can be applied to 2-SVM and to hard-SVM, whereas the geometric interpretation of 1-SVM is RCH-NPP, which is a much harder problem to solve (especially if we want to use kernels and the only operations available between patterns are inner products). Besides, 1-SVM can by no means be transformed into an instance of hard-SVM. However, 1-SVM has been claimed to be better than 2-SVM when redundant noise features are present [28].

Regarding the parameter values, it is clear that an optimal choice for C and σ for 1-SVM would be suboptimal for 2-SVM, and vice versa. Therefore, it is not possible to fix their values in advance as in the previous experiments. To overcome this, the values for 1-SVM are the ones that [24] uses, which are shown to result in pretty good models. As for the 2-SVM case, we have not been able to find a paper with suggestions for the datasets at hand, so we use here the results obtained in [29] after running the well-known evolutionary strategy CMA-ES [30] (whose publicly available code can be downloaded from [31]) over 50 generations. The function to be optimized by CMA-ES is the average test error over a 5-fold cross-validation of the 5 first training partitions of the datasets. This is a similar function to the one used to get the parameter values in [24], but take into account that we perform a continuous search in the parameter space with CMA-ES, whereas Rätsch carries out a discrete one by means of a grid. The final parameter choices are shown in table 5.3.1.

Let us now describe 1-SMO. With a similar reasoning to that for 2-SMO [7] it is easy to arrive to the following equations:

Dataset	C 1-SVM	σ 1-SVM	C 2-SVM	σ 2-SVM
Titanic	5.00	2.00	0.46	35.08
Heart	3.16	120.00	0.41	77.39
Diabetes	1.00	20.00	34.72	620.20
Cancer	15.19	50.00	9.90	62.48
Thyroid	10.00	3.00	0.82	1.00
Flare	1.02	30.00	4.23	2558.07
Splice	1000.00	70.00	114.69	50.44
Image	500.00	30.00	742.73	28.24
German	3.16	55.00	780.4	1443.53
Banana	316.20	1.00	2.60	1.40

Table 5.3.1: C and σ values used in the comparison between the models obtained for the 1-SVM and 2-SVM formulations.

$$\mathbf{x_L} = argmin_{\mathbf{x_i}\ |\ i\ \in\ SVNUB \cup (SVUB \cap I_2) \cup (NSV \cap I_1)} \left\{ m\left(\mathbf{x_i}\right) \right\}, \tag{5.3.1}$$

$$\mathbf{x_U} = argmax_{\mathbf{x_i}\ |\ i\ \in\ SVNUB \cup (SVUB \cap I_1) \cup (NSV \cap I_2)} \left\{ m\left(\mathbf{x_i}\right) \right\}, \tag{5.3.2}$$

where $m\left(\mathbf{x_i}\right)$ is defined as in (5.2.1). The other difference is that now we have to comply with the constraints $\alpha_i \leq C$. Thus, λ has to be cut always looking at two different quantities apart from itself, depending on which one of the four cases we are coping with. These two differences give then rise to the following algorithm:

Next we discuss which stopping criteria to use. Since we have not changed the code for SVM-Light the choice is obviously influenced by what this algorithm does; recall from §3.1.2 that it checks for the relaxed 1-SVM KKT conditions (3.1.1)–(3.1.3). Therefore, for the sake of similarity, CH-MDM has to be stopped when $\Delta \leq 2\epsilon$, with Δ defined in (4.1.3), since we showed in §4.1.3 that, by doing this, it is guaranteed that the relaxed 2-SVM KKT conditions are fulfilled with an absolute tolerance ϵ, that is, the same that SVM-Light does for 1-SVM.

Moreover, it is not difficult to show that stopping 2-SMO and 1-SMO with the same criterion $\Delta \leq 2\epsilon$, but with Δ defined in (5.2.4), has the same consequence. We consider here the 2-SMO case. From the definition of Δ and equations (5.2.1)–(5.2.3) we have:

$$\Delta = \mathbf{w} \cdot \left(\mathbf{x_U} - \mathbf{x_L}\right) - \left(y_U - y_L\right)$$
$$= \max_{i\ |\ i\ \in\ SV \cup (NSV \cap I_2)} \left\{ \mathbf{w} \cdot \mathbf{x_i} - y_i \right\} - \min_{i\ |\ i\ \in\ SV \cup (NSV \cap I_1)} \left\{ \mathbf{w} \cdot \mathbf{x_i} - y_i \right\}.$$

Since the relaxed KKT conditions can be summarized in the following equation [7]:

Algorithm 3 1-SMO algorithm for binary SVMs

```
1: Choose arbitrarily $L, U$ so that $y_L \neq y_U$ and set $\alpha_L = \alpha_U = 1, \alpha_i = 0\ \forall i \neq L, U$.
2: Calculate $A, B, C$ as well as $D_i, E_i\ \ \forall i$.
3: loop
4:     Set $\mathbf{x_L} = argmin_{\mathbf{x_i} \mid i \in SVNUB \cup (SVUB \cap I_2) \cup (NSV \cap I_1)} \{m(\mathbf{x_i})\}$.
5:     Set $\mathbf{x_U} = argmax_{\mathbf{x_i} \mid i \in SVNUB \cup (SVUB \cap I_1) \cup (NSV \cap I_2)} \{m(\mathbf{x_i})\}$.
6:     Set $\Delta = m(\mathbf{x_U}) - m(\mathbf{x_L})$.
7:     if stop criterion is true then
8:         Return $\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \mathbf{x_i},\ b = \frac{1}{|SV|} \sum_{i \in SV} (y_i - \mathbf{w} \cdot \mathbf{x_i})$.
9:     end if
10:    Set $F = \|\mathbf{x_L} - \mathbf{x_U}\|^2, \lambda = \frac{\Delta}{F}$.
11:    if $U, L \in I_1$ then
12:        $\lambda \Leftarrow \min\{\lambda, C - \alpha_L, \alpha_U\}$
13:        $A \Leftarrow A + \lambda(2(D_L - D_U) + \lambda F),\ C \Leftarrow C + \lambda(E_L - E_U)$
14:        $D_i \Leftarrow D_i + \lambda(\mathbf{x_i} \cdot \mathbf{x_L} - \mathbf{x_i} \cdot \mathbf{x_U})\ \ \forall i$
15:        $\alpha_L \Leftarrow \alpha_L + \lambda,\ \alpha_U \Leftarrow \alpha_U - \lambda$
16:    else if $U, L \in I_2$ then
17:        $\lambda \Leftarrow \min\{\lambda, \alpha_L, C - \alpha_U\}$
18:        $B \Leftarrow B + \lambda(2(E_U - E_L) + \lambda F),\ C \Leftarrow C + \lambda(D_U - D_L)$
19:        $E_i \Leftarrow E_i + \lambda(\mathbf{x_i} \cdot \mathbf{x_U} - \mathbf{x_i} \cdot \mathbf{x_L})\ \ \forall i$
20:        $\alpha_L \Leftarrow \alpha_L - \lambda,\ \alpha_U \Leftarrow \alpha_U + \lambda$
21:    else if $U \in I_2, L \in I_1$ then
22:        $\lambda \Leftarrow \min\{\lambda, C - \alpha_L, C - \alpha_U\}$
23:        $A \Leftarrow A + \lambda(2D_L + \lambda \mathbf{x_L} \cdot \mathbf{x_L}),\ B \Leftarrow B + \lambda(2E_U + \lambda \mathbf{x_U} \cdot \mathbf{x_U}),\ C \Leftarrow C + \lambda(E_L + D_U + \lambda \mathbf{x_L} \cdot \mathbf{x_U})$
24:        $D_i \Leftarrow D_i + \lambda \mathbf{x_i} \cdot \mathbf{x_L}\ \ \forall i,\ E_i \Leftarrow E_i + \lambda \mathbf{x_i} \cdot \mathbf{x_U}\ \ \forall i$
25:        $\alpha_L \Leftarrow \alpha_L + \lambda,\ \alpha_U \Leftarrow \alpha_U + \lambda$
26:    else
27:        $\lambda \Leftarrow \min\{\lambda, \alpha_L, \alpha_U\}$
28:        $A \Leftarrow A + \lambda(-2D_U + \lambda \mathbf{x_U} \cdot \mathbf{x_U}),\ B \Leftarrow B + \lambda(-2E_L + \lambda \mathbf{x_L} \cdot \mathbf{x_L}),\ C \Leftarrow C + \lambda(-E_U - D_L + \lambda \mathbf{x_L} \cdot \mathbf{x_U})$
29:        $D_i \Leftarrow D_i - \lambda \mathbf{x_i} \cdot \mathbf{x_U}\ \ \forall i,\ E_i \Leftarrow E_i - \lambda \mathbf{x_i} \cdot \mathbf{x_L}\ \ \forall i$
30:        $\alpha_L \Leftarrow \alpha_L - \lambda,\ \alpha_U \Leftarrow \alpha_U - \lambda$
31:    end if
32: end loop
```

$$\max_{i \mid i \in SV \cup (NSV \cap I_2)} \{\mathbf{w} \cdot \mathbf{x_i} - y_i\} - \epsilon \leq b \leq \min_{i \mid i \in SV \cup (NSV \cap I_1)} \{\mathbf{w} \cdot \mathbf{x_i} - y_i\} + \epsilon,$$

it is clear that in order to fulfil this we need $\Delta \leq 2\epsilon$. Thus, we have a criterion which is equivalent in spirit for the four algorithms (but not very convenient, as will be seen soon). As before, we compare them in terms of error percentage, number of support vectors and number of iterations with two different tolerances $\epsilon = 0.001$ and $\epsilon = 0.00001$. The results for $\epsilon = 0.001$ are given in tables 5.3.2, 5.3.3 and 5.3.4.

Several aspects can be pointed out from these three tables. First, it is clear that for some datasets such as *diabetes*, *image* and *german* the 0.001 accuracy is very inaccurate for CH-MDM, since it gives bad error rates and too few iterations and SVs while compared to 2-SMO. Nevertheless, for 2-SMO, 1-SMO and SVM-Light it seems to be accurate enough: observe that their error rates are similar for all the datasets, and at least as low as those of CH-MDM.

Secondly, the parameter choice is quite convenient, because comparing the rates of 2-SMO with those in table 5.2.2, they are better for most of the cases (sometimes they are slightly worse, but this can be due to the different stopping criterion).

Thirdly, the observation of the previous section that CH-MDM gives less SVs than 2-SMO is still valid, even for the datasets in which CH-MDM does not stop too early, whereas it is interesting to see that now 2-SMO always carries out more iterations than CH-MDM, which seems to contradict the impression we had that 2-SMO should be faster because of its being less restricted. However, this fact is related to the CH-MDM's inaccuracy we observed above: the vectors $\mathbf{w^{(t)}}$ are different in 2-SMO and CH-MDM, so an absolute stopping criterion such as the one we are using gives different results, as it is less demanding for one method than for the other one. In this particular case, for 2-SMO it is much more demanding than for CH-MDM. Unless ϵ is small enough, an absolute stopping criterion is subject to early stopping and is case-dependent. Therefore, a relative criterion should be used if possible.

Furthermore, we obtain very similar results for 1-SMO and SVM-Light in the three measures, as expected. Interestingly, 1-SMO is slightly better than SVM-Light, since it has the same accuracies, but gives a bit less support vectors and requires a few less iterations. Without having delved into the details of the SVM-Light code, we think that this fact is due to SVM-Light using an interior-point solver [15] for the general case where q is even. In the particular case of $q = 2$, this solver is less accurate than 1-SMO because it may not solve the task analytically.

Last but not least, comparing 1-SMO and SVM-Light with the other two algorithms, their error rates are similar and they obtain models with many less SVs than both 2-SMO and CH-MDM (CH-MDM gives less SVs only in the datasets for which it stops too early, whereas 2-SMO always gives more SVs than 1-SMO and SVM-Light). In terms of iterations, sometimes they are faster than 2-SMO, whereas sometimes it is the other way round. The comparison with CH-MDM is left for the more accurate tolerance.

Dataset	ERR. MDM	ERR. 2-SMO	ERR. 1-SMO	ERR. SVM-LIGHT
Titanic	22.8 ± 1.2	22.8 ± 1.2	22.4 ± 1.0^{3}	22.4 ± 1.0^{3}
Heart	15.8 ± 3.3	15.8 ± 3.3	15.9 ± 3.2	15.9 ± 3.2
Diabetes	39.1 ± 9.9	$23.2\pm1.7^{1,3}$	23.5 ± 1.7	23.5 ± 1.7
Cancer	27.5 ± 5.0	26.5 ± 4.9^{1}	26.3 ± 4.6	26.3 ± 4.6
Thyroid	4.3 ± 1.9	4.4 ± 1.9	4.4 ± 2.2	4.4 ± 2.2
Flare	44.4 ± 3.2	33.5 ± 1.7^{1}	32.8 ± 1.7^{3}	32.8 ± 1.7^{3}
Splice	10.7 ± 0.7	10.6 ± 0.7^{3}	10.8 ± 0.6	10.8 ± 0.6
Image	16.3 ± 10.8	$2.9\pm0.5^{1,3}$	3.1 ± 0.5	3.1 ± 0.5
German	44.4 ± 8.6	23.6 ± 2.1^{1}	23.6 ± 2.2	23.6 ± 2.2
Banana	10.4 ± 0.5^{3}	10.4 ± 0.5^{3}	11.6 ± 0.7	11.6 ± 0.7

Table 5.3.2: Average test accuracies given by the CH-MDM, 2-SMO, 1-SMO and SVM-Light algorithms, with $\epsilon = 0.001$. A superindex stands for a statistically significant difference in the paired samples; ^{1} for those of CH-MDM and 2-SMO, ^{2} for those of 1-SMO and SVM-Light, and ^{3} for those of the best one between CH-MDM and 2-SMO and the best one between 1-SMO and SVM-Light.

Dataset	SVs MDM	SVs 2-SMO	SVs 1-SMO	SVs SVM-LIGHT
Titanic	150.0 ± 0.0	150.0 ± 0.0	$68.6\pm9.5^{2,3}$	68.9 ± 9.6
Heart	161.8 ± 2.9^{1}	163.3 ± 2.4	82.5 ± 5.4^{3}	82.5 ± 5.4^{3}
Diabetes	$80.1\pm4.1^{1,3}$	412.7 ± 7.7	264.9 ± 7.4	265.0 ± 7.3
Cancer	152.6 ± 5.9^{1}	179.4 ± 5.9	114.1 ± 6.1^{3}	114.1 ± 6.0^{3}
Thyroid	86.6 ± 3.2^{1}	87.4 ± 3.1	25.3 ± 5.7^{3}	25.3 ± 5.7^{3}
Flare	$321.3\pm20.8^{1,3}$	664.6 ± 0.8	477.1 ± 12.1^{2}	478.2 ± 12.1
Splice	$538.8\pm19.5^{1,3}$	727.8 ± 12.5	620.1 ± 14.4^{2}	620.7 ± 14.7
Image	$100.0\pm15.8^{1,3}$	215.3 ± 11.5	167.6 ± 9.1^{2}	172.1 ± 9.6
German	$16.6\pm2.9^{1,3}$	590.2 ± 12.5	407.6 ± 10.8	407.6 ± 10.8
Banana	211.4 ± 12.0^{1}	230.8 ± 14.1	89.6 ± 10.2^{3}	89.5 ± 10.2^{3}

Table 5.3.3: Average number of support vectors given by the CH-MDM, 2-SMO, 1-SMO and SVM-Light algorithms, with $\epsilon = 0.001$. A superindex stands for a statistically significant difference in the paired samples; ^{1} for those of CH-MDM and 2-SMO, ^{2} for those of 1-SMO and SVM-Light, and ^{3} for those of the best one between CH-MDM and 2-SMO and the best one between 1-SMO and SVM-Light.

In order to confirm or refute the appreciations of the results in the previous tables, we show next the results for $\epsilon = 0.00001$ in tables 5.3.5, 5.3.6 and 5.3.7.

Here we see that this new tolerance is accurate enough for CH-MDM, since very similar values appear in the error and SV columns for CH-MDM and 2-SMO, like in table 5.2.2. Clearly both methods are converging to the same model (with the only exception of dataset *german*, where even this ϵ does not seem to be enough for CH-MDM. As was stated above, the accuracy of a given ϵ is case-dependent whenever an absolute criterion is used.). 1-SMO and SVM-Light are also converging to the same model (which is the same than the one in the previous tables, since the values are almost identical), and 1-SMO keeps on being slightly better than its counterpart.

It is also evident that not only the values for 1-SMO and SVM-Light in tables 5.3.2, 5.3.5, 5.3.3 and 5.3.6 are nearly identical, but also for 2-SMO, so this confirms our suspicion that $\epsilon = 0.001$ is accurate enough for these three methods. For SVM-Light Joachims also concluded this fact in [15], the only effect of reducing ϵ being a remarkable increase in the number of iterations, as we can see if we compare tables 5.3.4 and 5.3.7.

The parameter choices for 2-SVM give better error rates that those in table 5.2.2 (except for datasets *titanic* and *flare*), so these choices made by CMA-ES are not optimal but remarkably good, because the rates are similar to those obtained for 1-SVM with the values suggested by [24].

Comparing the speed of 2-SMO and CH-MDM, in table 5.3.7 it looks like CH-MDM is faster, as we mentioned for table 5.3.4. However, this comparison is not fair, because 2-SMO with $\epsilon = 0.001$ achieves a similar accuracy than CH-MDM with $\epsilon = 0.00001$. If we compare then, column 3 of table 5.3.4 and column 2 of table 5.3.7, it is not clear which of the algorithms is faster: CH-MDM wins for datasets *diabetes*, *cancer*, *splice*, *image*, *german* and *banana*, whereas SMO wins for *titanic*, *heart*, *thyroid* and *flare* (take into account that for *german* and perhaps for *image* ϵ should be lower for CH-MDM). However, several facts can influence in these results: it is possible that with $\epsilon > 0.001$ 2-SMO still gives good results for the datasets where CH-MDM seems to beat it. Besides, it may happen that CH-MDM arrives to two points which are not the closest ones in the hulls, but that nonetheless form by chance a $\mathbf{w}$ with the proper orientation, causing that the KKT conditions are fulfilled in advance.

Finally, comparing 1-SMO and SVM-Light versus the other two, table 5.3.6 confirms that 1-SVM manages to give models with less SVs than 2-SVM. Moreover, looking at table 5.3.7 for its iterations, it also turns out to be in general faster than 2-SMO, but not always (note that the datasets for which it is slower are the same that those in table 5.3.4).

Dataset	IT. MDM	IT. 2-SMO	IT. 1-SMO	IT. SVM-LIGHT
Titanic	228.1±8.7^{1}	284.6±17.1	144.3±21.8^{2,3}	148.7±25.0
Heart	214.7±7.3^{1}	253.5±8.8	149.5±30.9^{2,3}	154.0±33.6
Diabetes	78.3±4.2^{1,3}	1255.4±32.3	359.5±54.0^{2}	374.7±55.4
Cancer	194.3±8.4^{1,3}	887.4±42.1	1434.8±403.3^{2}	1485.5±395.9
Thyroid	130.0±4.7^{1,3}	158.7±8.0	213.0±73.9^{2}	225.7±76.8
Flare	319.4±20.8^{1,3}	1158.3±42.5	615.3±153.3^{2}	638.8±192.8
Splice	559.1±16.2^{1,3}	2716.1±323.5	2017.8±161.5	2032.6±116.2
Image	194.7±43.0^{1,3}	36498.9±2344.3	56239.3±13048.2^{2}	61237±13282.8
German	15.3±3.2^{1,3}	19462.7±665.4	1359.5±102.9^{2}	1394.8±104.7
Banana	318.5±16.6^{1,3}	991.7±66.3	41899.8±17406.6^{2}	44201.9±19379.7

Table 5.3.4: Average number of iterations given by the CH-MDM, 2-SMO, 1-SMO and SVM-Light algorithms, with $\epsilon = 0.001$. A superindex stands for a statistically significant difference in the paired samples; ^{1} for those of CH-MDM and 2-SMO, ^{2} for those of 1-SMO and SVM-Light, and ^{3} for those of the best one between CH-MDM and 2-SMO and the best one between 1-SMO and SVM-Light.

Dataset	ERR. MDM	ERR. 2-SMO	ERR. 1-SMO	ERR. SVM-LIGHT
Titanic	22.8±1.2	22.8±1.2	22.4±1.0^{3}	22.4±1.0^{3}
Heart	15.8±3.3	15.8±3.3	15.9±3.2	15.9±3.2
Diabetes	23.2±1.7^{3}	23.2±1.7^{3}	23.5±1.7	23.5±1.7
Cancer	26.5±4.9	26.5±4.9	26.3±4.6	26.3±4.6
Thyroid	4.4±1.9	4.4±1.9	4.4±2.2	4.4±2.2
Flare	33.5±1.7	33.5±1.7	32.8±1.7^{3}	32.8±1.7^{3}
Splice	10.6±0.7^{3}	10.6±0.7^{3}	10.8±0.6	10.8±0.6
Image	3.0±0.5	2.9±0.5^{3}	3.1±0.5	3.1±0.5
German	27.7±3.5	23.6±2.1^{1}	23.6±2.2	23.6±2.2
Banana	10.4±0.5^{3}	10.4±0.5^{3}	11.6±0.7	11.6±0.7

Table 5.3.5: Average test accuracies given by the CH-MDM, 2-SMO, 1-SMO and SVM-Light algorithms, with $\epsilon = 0.00001$. A superindex stands for a statistically significant difference in the paired samples; ^{1} for those of CH-MDM and 2-SMO, ^{2} for those of 1-SMO and SVM-Light, and ^{3} for those of the best one between CH-MDM and 2-SMO and the best one between 1-SMO and SVM-Light.

Dataset	SVs MDM	SVs 2-SMO	SVs 1-SMO	SVs SVM-LIGHT
Titanic	150.0±0.0	150.0±0.0	68.6±9.5^{2,3}	69.1±9.6
Heart	163.4±2.4	163.4±2.4	82.5±5.4^{3}	82.5±5.4^{3}
Diabetes	407.2±7.4^{1}	412.8±7.7	264.9±7.3^{3}	264.9±7.3^{3}
Cancer	179.2±5.9^{1}	179.4±5.9	113.9±6.1^{2,3}	114.1±6.2
Thyroid	87.4±3.1	87.4±3.1	25.3±5.7^{3}	25.3±5.7^{3}
Flare	664.6±0.7	664.6±0.8	477.4±12.1^{2,3}	481.0±13.0
Splice	725.7±12.3^{1}	728.9±12.9	620.3±14.2^{2,3}	621.3±14.6
Image	218.6±11.4	215.3±11.5^{1}	167.3±9.2^{2,3}	172.6±9.6
German	498.1±14.5^{1}	590.1±12.4	407.7±10.8^{3}	407.7±10.8^{3}
Banana	230.7±14.2^{1}	230.9±14.1	89.4±10.1^{3}	89.4±10.1^{3}

Table 5.3.6: Average number of support vectors given by the CH-MDM, 2-SMO, 1-SMO and SVM-Light algorithms, with $\epsilon = 0.00001$. A superindex stands for a statistically significant difference in the paired samples; ^{1} for those of CH-MDM and 2-SMO, ^{2} for those of 1-SMO and SVM-Light, and ^{3} for those of the best one between CH-MDM and 2-SMO and the best one between 1-SMO and SVM-Light.

Dataset	IT. MDM	IT. 2-SMO	IT. 1-SMO	IT. SVM-LIGHT
Titanic	367.4±15.6^{1}	423.6±22.2	192.4±31.5^{2,3}	196.6±37.2
Heart	365.3±12.8^{1}	402.0±16.2	269.1±87.8^{3}	271.3±85.8^{3}
Diabetes	755.1±17.7^{1}	2046.1±50.2	602.6±146.3^{2,3}	616.4±144.0
Cancer	748.5±30.8^{1,3}	1503.6±68.3	3534.5±1844.6	3608.6±1803.9
Thyroid	226.4±8.6^{1,3}	258.7±12.0	412.1±170.2^{2}	426.3±169.8
Flare	1092.9±29.3^{1,3}	1790.3±73.4	1414.3±1106.2	1489.4±1256.7
Splice	2136.9±180.1^{1,3}	7287.6±1134.2	3697.4±340.2	3713.3±277.1
Image	8547.6±377.5^{1,3}	65610.0±4857.0	136723.3±27906.2^{2}	158734.5±36668.3
German	1569.5±80.0^{1,3}	32809.9±1164.5	2417.6±272.8^{2}	2463.0±271.3
Banana	979.0±59.8^{1,3}	1755.0±112.9	99563.5±76471.3^{2}	114301.4±89833.6

Table 5.3.7: Average number of iterations given by the CH-MDM, 2-SMO, 1-SMO and SVM-Light algorithms, with $\epsilon = 0.00001$. A superindex stands for a statistically significant difference in the paired samples; ^{1} for those of CH-MDM and 2-SMO, ^{2} for those of 1-SMO and SVM-Light, and ^{3} for those of the best one between CH-MDM and 2-SMO and the best one between 1-SMO and SVM-Light.

Chapter 6

Discussion and Additional Work

Here we give a concise discussion of the results obtained in the previous chapters. Afterwards, a synopsis of the papers and articles that have been written before or at the time of writing this work is presented. Finally, we point out some possible future lines of action to keep on with this research.

6.1 Discussion

We have shown experimentally in the previous chapter the main theoretical results in §4. In summary, we have seen the following to be true:

- **2-SVMs:** the adaptation of Keerthi's SMO to quadratic penalty SVMs (2-SMO) operates in the dual analogously to the way CH-MDM operates in the CH-NPP. In the end, both methods converge to the same model, but in a slightly different way: CH-MDM removes support vectors more quickly, whereas 2-SMO converges faster, since it is less restricted in the choice of the working set.
- **1-SVMs:** Keerthi's SMO (1-SMO), which operates on linear penalty SVMs, operates in the same way than SVM-Light with working sets of just two patterns. 1-SMO has a slight advantage in comparison since it solves analytically each iteration's task. Therefore, it converges a bit faster than its counterpart. The shrinking heuristic in SVM-Light does not make any difference for the parameters and datasets used.
- **One against another:** both SVM paradigms achieve similar performances if their parameters are conveniently chosen. Besides, 1-SVM yields models with less support vectors than 2-SVM ones, and generally the convergence is faster. However, no definite conclusions can be drawn, as another choice of parameters may well result

in similar accuracies, but in a different number of support vectors and a different convergence speed.

Thus, CH-MDM has been put into relationship with SMO and SVM-Light, and also it has been used as an intermediary to put into relationship the latter two algorithms. Moreover, in §4 it was not only considered as a decomposition method, but also as a feasible direction one (while choosing the patterns to update). Hence, our observations open a new way that we think is worth exploring: basically, we are saying that decomposition and geometrical algorithms are not very different all in all, so that results in one of these two trends can be used in the other trend. Besides, there are algorithms in both trends that use common auxiliary optimization techniques (such as feasible direction ones), so contributions and ideas in these auxiliary techniques can be used to improve SVM training algorithms too.

For instance, we have seen how taking the geometrical MDM algorithm, which was originally designed with a different objective in mind than training SVMs, can be easily adapted to this task. It has been seen as well how the three algorithms considered apply either implicitly (SMO and CH-MDM) or explicitly (SVM-Light) Zoutendijk's feasible direction method to choose the patterns for solving three different versions of the SVM training task: linear dual (1-SMO and SVM-Light), quadratic dual (2-SMO) and CH-NPP (CH-MDM).

6.2 Additional Work

6.2.1 Previous and Simultaneous Work

This new way has begun to be explored soon after coming up with the observations that compose this work. The following papers have arisen as a result:

- In [32] a different and shorter version of this work's results is presented. Additionally, we suggest a possibility for an RCH-MDM algorithm, which basically consists of restricting 1-SMO to choose patterns from the same class, and constraining its updates to lie into the associated reduced convex hull.

- [33] presents an extension of CH-MDM to work with four patterns at the same time, that is, in the two hulls simultaneously. We suspect that this algorithm can be put into relationship with SVM-Light working with $q = 4$, as well as with a hypothetical SMO that chooses four patterns to update in each iteration.

- [34] explains how CH-MDM can be inefficient in some cases where cycles in the update directions appear. We suggest a version of the algorithm that checks for cycles and combines the update directions in a cycle to try to get out of it. This version is shown to be at least as good as the normal one, and normally it is faster (always converging to the same model than CH-MDM).

Here is a list of somewhat less related papers:

- In [35] and its extended version [29] we give an algorithm for parameter optimization that gives quite good results for low dimensionality, and show its application to find the C and σ parameters of SVMs for binary classification.
- [36] covers Support Vector Regression (SVR), a technique that is not considered in this work. It compares the performance of 1-SVMs and 2-SVMs for this task, in a pretty similar way than we did in §5.3. Besides, an alternative proof that CH-NPP is a rescaled reformulation of the 2-SVM dual is given.
- [37] describes an application of Support Vector Regression to predict the daily wind energy production of a subset of the wind farms located in Spain.

6.2.2 Future Work

To conclude, we enumerate possible ideas and lines of action that we are working on now or plan to explore in the near future:

- Compare our RCH-MDM suggestion in [32] with the one given in [21]. We think that ours will be faster, since it again applies implicitly Zoutendijk's method to find the steepest first order descent direction, whereas Tao's is a somewhere in-the-middle approach.
- Analyze whether cycles in working set selection also appear in SMO like in CH-MDM. If that is the case, we would like to accelerate SMO in the same way that we did with CH-MDM in [34].
- Explore the applicability of second order working set selection in CH-MDM as has been done for SMO [8, 38]. Check whether second order has or has not a specific geometrical and/or a feasible direction sense.
- Analyze SVR, since recently a geometrical interpretation of it has appeared in [19], as well as some SMO proposals to solve the inherent task [39]. We think that it is

possible to apply similar reasonings in this case to the ones we have been discussing for the classification case.

- Study the ν-SVM paradigm [40], since it gets rid of the C parameter substituting it with a ν bounded parameter. Besides, ν-SVM classification has been shown to be equivalent to RCH-NPP [12].

Bibliography

[1] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[2] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 2002.

[3] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A(209):415–446, 1909.

[4] B.F. Mitchell, V.F. Dem'yanov, and V.N. Malozemov. Finding the point of a polyhedron closest to the origin. *SIAM J. Contr.*, 12:19–26, 1974.

[5] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, 2000.

[6] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Machines*, pages 185–208, 1999.

[7] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murphy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[8] R.E. Fan, P.H. Chen, and C.J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[9] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, Cambridge, 2000.

[10] T.T. Friess. Support vector networks: the kernel adatron with bias and soft-margin. *Univ. Sheffield Tech. Rep.*, 1998.

[11] K.P. Bennett and E.J. Bredensteiner. Duality and geometry in svm classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 57–64, 2000.

[12] D.J. Crisp and C.J.C. Burges. A geometric interpretation of ν-svm classifiers. *Adv. Neural Inform. Process. Syst. (NIPS)*, 12:244–250, 1999.

[13] V. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer-Verlag, Berlin, 1982.

[14] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. *Proc. 1997 IEEE Workshop*, pages 276–285, 1997.

[15] T. Joachims. Making large-scale support vector machine learning practical. *Advances in Kernel Methods - Support Vector Machines*, pages 169–184, 1999.

[16] G. Zoutendijk. *Methods of Feasible Directions: a Study in Linear and Non-linear Programming.* Elsevier, 1970.

[17] E.G. Gilbert. Minimizing the quadratic form on a convex set. *SIAM J. Contr.*, 4:61–79, 1966.

[18] V. Franc and V. Hlaváč. An iterative algorithm learning the maximal margin classifier. *Pattern Recognition*, 36:1985–1996, 2003.

[19] J. Bi and K.P. Bennett. A geometric approach to support vector regression. *Neurocomputing*, 55:79–108, 2003.

[20] M.E. Mavroforakis and S. Theodoridis. A geometric approach to support vector machine (svm) classification. *IEEE Transactions on Neural Networks*, 17(3):671–682, May 2006.

[21] Q. Tao, G. W. Wu, and J. Wang. A general soft method for learning svm classifiers with l1-norm penalty. *Pattern Recognition*, 41:939–948, 2008.

[22] M. Bazaraa, D. Sherali, and C. Shetty. *Nonlinear Programming: Theory and Algorithms.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, 1992.

[23] R. Collobert and S. Bengio. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[24] G. Rätsch. Benchmark repository.
http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm.

[25] University of California Irvine. Uci machine learning repository.
http://archive.ics.uci.edu/ml.

[26] The r language for statistical computing.
http://www.r-project.org.

[27] T. Joachims. Svm-light support vector machine.
http://svmlight.joachims.org.

[28] J. Zhu, T. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[29] Á. Barbero, J. López, and J. Dorronsoro. Finding optimal model parameters by discrete grid search. Submitted to Neurocomputing in February 2008.

[30] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[31] N. Hansen. The cma evolution strategy: Source code.
http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html.

[32] J. López, Á. Barbero, and J. Dorronsoro. On the equivalence of the smo and mdm algorithms for svm training. Submitted to the European Conference on Machine Learning (ECML 2008).

[33] Á. Barbero, J. López, and J. Dorronsoro. A 4-vector mdm algorithm for support vector training. In *Proceedings of the 18th International Conference in Artificial Neural Networks (ICANN)*. Springer, 2008 (por aparecer).

[34] Á. Barbero, J. López, and J. Dorronsoro. An accelerated mdm algorithm for svm training. In *Proceedings of the 11th European Symposium on Artificial Neural Networks*, pages 421–426, 2008.

[35] A. Barbero, J. López, and J. Dorronsoro. Finding optimal model parameters by discrete grid search. In *Advances in Soft Computing: Innovations in Hybrid Intelligent Systems 44*, pages 120–127. Springer, 2008.

[36] Á. Barbero, J. López, and J. Dorronsoro. Square penalty support vector regression. In *Lectures Notes in Computer Science: Intelligent Data Engineering and Automated Learning - IDEAL 2007*, pages 537–546. Springer, 2007.

[37] Á. Barbero, J. López, and J. Dorronsoro. Kernel methods for wide area wind generation forecasting. In *Proceedings of the 2008 European Wind Energy Conference (EWEC)*, 2008 (por aparecer).

[38] T. Glasmachers and Ch. Igel. Second order smo improves svm online and active learning. *Neural Computation*, 20(2):374–382, 2008.

[39] B. Schölkopf and A. J. Smola. *Learning With Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* Machine Learning. MIT Press, 2002.

[40] B. Schölkopf and A. J. Smola. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.