UNIVERSIDAD AUTÓNOMA DE MADRID

Analysis and Convergence of SMO-like Decomposition and Geometrical Algorithms for Support Vector Machines

by

Jorge López Lázaro

under the supervision of

José Ramón Dorronsoro Ibero

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Escuela Politécnica Superior

Departamento de Ingeniería Informática

November 2011

"Being a PhD student is like becoming all of the Seven Dwarves. In the beginning you're Dopey and Bashful. In the middle, you are usually sick (Sneezy), tired (Sleepy), and irritable (Grumpy). But at the end, they call you Doc, and then you're Happy"

Ronald T. Azuma

Abstract

Support Vector Machines (SVMs) constitute one of the most successful paradigms in Machine Learning nowadays. Their success stems from the fact that they are relatively simple models, with excellent generalization properties for classification and regression, that arise from solving convex optimization problems. In addition, the models are interpretable in terms of the so–called Support Vectors, which are the points that influence in the final models.

These models have the form of a hyperplane, so SVMs are a variety of linear models. Even though linear models are in principle of limited use, the power behind SVMs comes from the use of kernel functions, which effectively build non–linear models after arriving to a linear model in a projected feature space. The success of SVMs is also indicated by the formulation over the years of numerous variations building on them. Among these, ν–SVMs and Least Squares SVMs (LS–SVMs) have been especially relevant. A parallel line of research investigates the geometrical formulation of SVMs as Nearest Point Problems.

Although the optimization problems giving rise to SVMs have a simple structure, it is not trivial to solve efficiently these tasks. The main problem comes from the size of the kernel matrix, which is the square of the number of training patterns. This precludes the use of standard optimization routines, and requires the conception of ad–hoc methods. Perhaps the simplest method of all is Sequential Minimal Optimization (SMO). Despite its simplicity, some variations from the original algorithm, termed jointly as "SMO–like" methods, can be considered as the state–of–the–art in SVM training.

In this thesis, after motivating theoretically SVMs and the SMO algorithm, we formulate a general problem that encompasses all the specific formulations enumerated above. The SMO algorithm can be adapted to this general problem after minor changes, which also includes as particular cases the SMO variants for the different formulations. Moreover, we give a new and simple proof of the convergence of this general SMO version to the optimal solution.

Resumen

Las Máquinas de Vectores de Soporte (SVMs) constituyen hoy en día uno de los paradigmas más exitosos dentro del Aprendizaje Automático. Su éxito radica en el hecho de que son modelos relativamente simples, con propiedades excelentes de generalización para clasificación y regresión, los cuales resultan de resolver problemas de optimización convexa. Además, los modelos son interpretables en términos de los que se conocen como Vectores de Soporte, que son los puntos que influyen en los modelos finales.

Estos modelos tienen la forma de un hiperplano, por lo que las SVMs son una variedad de modelos lineales. A pesar de que los modelos lineales tienen un dominio de aplicación limitado, la potencia de las SVMs proviene del uso de funciones de núcleo, que en realidad construyen modelos no lineales a partir de modelos lineales en el espacio de características adonde proyectan. El éxito de las SVMs también queda patente con las numerosas variantes basadas en ellas que se han ido formulando a lo largo de los años. Entre ellas, las Máquinas de Vectores de Soporte ν (ν–SVMs) y las Máquinas de Vectores de Soporte de Mínimos Cuadrados (LS–SVMs) han tenido especial relevancia. Una línea paralela de investigación investiga la formulación geométrica de las SVMs como Problemas de Puntos Más Cercanos.

Aunque los problemas de optimización que originan las SVMs tienen una estructura simple, no es una cuestión trivial el resolverlos de manera eficiente. El principal problema es el tamaño de la matriz de núcleos, que es del cuadrado del número de patrones de entrenamiento. Esto hace imposible el uso de rutinas estándar de optimización, y requiere el diseño de métodos ad–hoc. Quizás el método más sencillo de todos sea la Optimización Secuencial Mínima (SMO). A pesar de su simplicidad, algunas variantes del problema original, denominadas en conjunto métodos "a la SMO", pueden considerarse como el estado del arte en el entrenamiento de SVMs.

En esta tesis, después de motivar de manera teórica las SVMs y el algoritmo SMO, formulamos un problema general que incluye todas las formulaciones específicas enumeradas arriba. El algoritmo SMO puede adaptarse a este problema tras unos pequeños cambios, lo cual también incluye como casos particulares las variantes SMO específicas de las distintas formulaciones. Además, damos una demostración nueva y sencilla de la convergencia a la solución óptima de esta versión general de SMO.

Acknowledgements

First of all, I am especially indebted to my supervisor José R. Dorronsoro for introducing me to the fascinating world of Machine Learning and Pattern Recognition, as well as for his constant and steady guiding throughout my doctoral research. His comments and discussions were of invaluable help.

I express my gratitude to Professors Johan A.K. Suykens and Joos Vandewalle for giving me the opportunity to join their department ESAT within Katholieke Universiteit Leuven, during the summer stays of 2009 and 2010. I would also like to thank all the members of that department for their warmth towards me and for many interesting discussions and chats.

I would also like to thank the Spanish Ministry of Science and Innovation (MICINN) for their funding with an FPU fellowship, which has allowed me both to develop this work and to join the aforementioned Leuven team.

I am also immensely grateful to all the professors who volunteered to come to the thesis panel, as well as to Professor Sergios Theodoridis from National and Kapodistrian University of Athens and Ph.D. Kris De Brabanter from Katholieke Universiteit Leuven for accepting to supervise this dissertation.

Back in Spain, I wholly appreciate the friendship of my colleague Álvaro Barbero, my faithful companion in this adventure of Ph.D. studies, as well as his collaboration in many coauthored papers. Finally all this effort seems to come to an end!

Last but not least, I would like to thank all people here in Madrid: my family, to which this dissertation is dedicated, my friends from high school, college and other contexts (too many to be named here!), as well as all the staff from IIC and the Department of Computer Science at Universidad Autónoma de Madrid, for their personal, technical (and sometimes even financial) support.

Contents

List of Figures

List of Tables

Abbreviations

CH	**C**onvex **H**ull
ERM	**E**mpirical **R**isk **M**inimization
GSK	**G**ilbert–**S**chlesinger–**K**ozinec algorithm
KKT	**K**arush–**K**uhn–**T**ucker optimality conditions
LS	**L**east–**S**quares
MDM	**M**itchell–**D**em'yanov–**M**alozemov algorithm
MNP	**M**inimum **N**orm **P**roblem
MVP	**M**ost **V**iolating **P**air
NPP	**N**earest **P**oint **P**roblem
OC	**O**ne–**C**lass
QP	**Q**uadratic **P**rogramming
RCH	**R**educed **C**onvex **H**ull
RKHS	**R**eproducing **K**ernel **H**ilbert **S**pace
SMO	**S**equential **M**inimal **O**ptimization algorithm
SRM	**S**tructural **R**isk **M**inimization
SV	**S**upport **V**ector
SVC	**S**upport **V**ector **C**lassification
SVM	**S**upport **V**ector **M**achine
SVR	**S**upport **V**ector **R**egression
VC	**V**apnik–**C**hervonenkis

Notation

In general, upper-case letters denote matrices, whereas lower-case letters denote vectors (if used with $\overrightarrow{\cdot}$) or scalars (if used plainly). It should be clear from the context when this is not so; for example there are some exceptions like the scalar C, due to common use (refer to the next page for a detailed list). For a vector $\overrightarrow{v}$, v_i denotes its i–th entry (this should not be confused with $\overrightarrow{v}_i$, which denotes the i–th vector in an enumeration), whereas for a matrix A, A_{ij} denotes its entry in the i–th row and j–th column. Caligraphic font is used generally for spaces, index sets and abstract entities.

Regarding operators, standard notation is used: $\|\overrightarrow{v}\|$ means the norm of a vector $\overrightarrow{v}$, ∇ is used for the gradient, E for expectation, and $\langle \overrightarrow{u}, \overrightarrow{v} \rangle$ is the inner product between vectors u and v. The operator $\leftarrow$ will be used for assignation in algorithmic codes. $\mathcal{O}$ stands for computational cost.

As for superscripts, t will be related to time (t–th iteration), and sequences will be delimited with curly braces: for instance, $\{\overrightarrow{v}^t\}$ is the abbreviation for vectors $\overrightarrow{v}^0, \overrightarrow{v}^1, \ldots$. A superscript $*$ is used for both optimality and for the convex conjugate (it should be clear from the context which is which). The notation $\widetilde{\cdot}$ is used for variables in cases where the augmented feature space is being used. A $\prime$ is generally used for auxiliary vectors. Other symbols should be clear from the context.

Symbols

$\overrightarrow{\alpha}$	Dual vector of Lagrange coefficients
b	Bias term
C	Trade–off parameter in regularized risk
γ	Bias of the positive class in the (R)CH–Margin formulation
d	Dimension of patterns in the input space
$d_{\mathcal{H}}$	Dimension of patterns in the feature space
D	Diameter of a sample of patterns
$D_{\pm}$	Diameter of the patterns belonging to the positive or negative class
$\mathcal{D}$	Dual objective function
δ_{ij}	Kronecker's delta symbol: 1 if $i = j$, 0 otherwise
Δ	KKT violation
ϵ	Width of the insensitive region of a loss function
	An arbitrarily small positive number
η	Bias of the negative class in the (R)CH–Margin formulation
$\overrightarrow{\xi}$	Vector containing the slack variables
f	Decision function
$\mathcal{H}$	Feature space
$\mathcal{I}_{\pm}$	Indexes of patterns belonging to the positive or negative class
$\mathcal{I}_L$	Indexes of patterns that can be chosen as L
$\mathcal{I}_U$	Indexes of patterns that can be chosen as U
k	Kernel function: $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
K	Kernel matrix ($n \times n$)
l	Lower bound for the coefficients in the general formulation
l_p	Type of norm used, dependent on p
L	Index of the pattern with smallest margin chosen by SMO
$\mathcal{L}$	Lagrangian of an optimization problem
λ	Advance factor
μ	Reduction coefficient for reduced convex hulls
N	Number of patterns in training

ν	Trade–off parameter in ν–SVMs
p	Value for l_p–norms (usually 1 or 2)
$\mathcal{P}$	Primal objective function
Φ	Feature map $\Phi : \mathcal{X} \to \mathcal{H}$
$\overrightarrow{q}$	Auxiliary vector in the general formulation
ρ	Margin parameter in ν–SVMs, OC and the general formulation
σ	Width of the Gaussian kernel
$\overrightarrow{t}$	Label vector in the general formulation
$\mathcal{T}$	Training set
$\mathcal{T}_{test}$	Test set
$\overrightarrow{\tau}$	Auxiliary vector in the general formulation
θ	Dual function in an optimization problem
$\overrightarrow{w}$	Weight vector
Ω	Gram matrix in the general formulation $(n \times n)$: $\Omega_{ij} = t_i t_j K_{ij}$
u	Trade–off parameter in the general formulation
U	Index of the pattern with largest margin chosen by SMO
v	Upper bound for the coefficients in the general formulation
$\overrightarrow{x}$	A given pattern (just considering the input)
$\mathcal{X}$	Input space
$\overrightarrow{y}$	Label vector
$\mathcal{Y}$	Output space
$\overrightarrow{z}$	Difference of patterns in the feature space

To my family

Chapter 1

The Learning Problem

This chapter is intended to give the reader a general view of the field of Machine Learning, as well as what the general objective that we seek to solve in Pattern Recognition is. It also starts to describe the general framework and theory upon which this thesis is based, while familiarizing the reader with the specific terminology we will make use of in the rest of the thesis.

The material for this chapter is mainly adapted from the books [2] and [3], with occasional references to other works [4–7].

1.1 Machine Learning and Pattern Recognition

Machine Learning is a broad scientific discipline that deals with algorithms that make computers show an intelligent behavior after being given a number of data. These algorithms are used in numerous practical applications including, among others, Computer Vision, Natural Language Processing, Medical Diagnosis, Bioinformatics or Speech and Handwriting Recognition. Depending on what do we mean by "behavior", we can distinguish several fields:

- In *Adaptive Control* the computer seeks to decide and modify the control of a given system.
- In *Pattern Recognition* the computer assigns an output value for each data instance.
- In *Data Mining* the computer extracts trends from the available data.

The frontiers between these fields are sometimes fuzzy, due to their rapid development and extensions. What is more, Machine Learning is also closely related to other scientific disciplines such as Artificial Intelligence, Statistics, Computational Neuroscience and Theoretical Computer Science.

In this thesis, we will center on Pattern Recognition, so our aim will be assigning "reasonable" *labels* (also commonly called *outputs* or *targets*) to *patterns* of data (also *instances*, *inputs*, *samples* or *observations*). We will later formalize and characterize mathematically what do we mean by "reasonable" or "intelligent" labelling.

For now, we will stick to general ideas and common language. Given some data, a *learning algorithm* will generate an explanatory *model* for these data, and this model will be used to label unseen patterns. Ideally, this model should explain as accurately as possible the given data, and *generalize* as well as possible (that is, label as accurately as possible new or unseen data). Since these two objectives are usually conflicting, we will need to find a balance between them. The given data are usually called the *training data* (also *training set*), whereas the process of learning a model is usually referred to as *training* or *learning*.

Unless stated otherwise, the training set is henceforth assumed to contain patterns that are labelled correctly, be it by experts or by accurate measures. This is what is known as *Supervised Learning*, in contrast to *Unsupervised Learning*, where the learning algorithm tries to extract trends in the unlabelled training data so as to assign a reasonable output to unseen data.

Concerning the instances of input data, a pattern can be described as a vector of *features*. These features together constitute all known characteristics of a pattern. Without loss of generality, we will always assume that for a given pattern all its features are available (that is, they have a value). If this were not so, one possibility is to remove all incomplete patterns from the training set before the training phase. Some more sophisticated techniques for dealing with *missing data* are discussed for instance in [8].

As for the outputs, we can distinguish between two basic problems belonging to the field of Pattern Recognition: *classification* and *regression*. In the former case, the outputs are restricted to a discrete set. If there are only two possible values (typically ± 1), we have *binary classification*, whereas if there are more possible values we talk of *multiclass classification*. In the latter case, the outputs are continuous (typically real-valued), yielding a *regression* task. This work will be centred on binary classification and regression problems.

1.2 Supervised Learning

Once we have informally set the problem of learning in general terms, next we state this problem more formally. We consider two spaces: the input space $\mathcal{X}$ and the output space $\mathcal{Y}$. Patterns belong to space $\mathcal{X}$, whereas their targets lie in space $\mathcal{Y}$. Thus, we can define the training set in the following way

$$\mathcal{T} = \{(\overrightarrow{x}_1, y_1), \ldots, (\overrightarrow{x}_N, y_N)\} \in (\mathcal{X} \times \mathcal{Y})^N, \tag{1.1}$$

where N is the number of training patterns and operator $\times$ denotes the Cartesian product. Note that, since we are dealing with Supervised Learning, each input instance consists of the actual pattern $\overrightarrow{x}_i$ and its associated label y_i. Depending on how this training set is presented, we distinguish among two kinds of supervised learning: in *Batch Learning* all the patterns are presented at once, whereas in *Online Learning* they appear one by one. Unless stated otherwise, we assume throughout the rest of the work that we are dealing with Batch Learning.

The relationship between the input and output spaces is assumed to be given by a probability distribution $p(\overrightarrow{x}, y)$ over the set $\mathcal{X} \times \mathcal{Y}$. Although this distribution is unknown (if it were known, the relationship between the outputs and inputs would not need to be estimated), the training samples are assumed to have been drawn independently from this distribution. This is commonly referred in the literature as the training samples being *i.i.d.* (independent and identically distributed).

The goal of Supervised Learning is to learn a model that reflects as well as possible this probability distribution. This model will be characterized by a *decision function* $f^* : \mathcal{X} \to \mathcal{D}$ ($\mathcal{D}$ is the set of possible decisions, and it is assumed that $\mathcal{Y} \subseteq \mathcal{D}$) that assigns an output $f^*(\overrightarrow{x})$ to a given pattern $\overrightarrow{x}$. This function f^* is chosen from the space $\mathcal{F}$ of all admissible functions, termed *hypotheses space*. The question now is how we can compare a decision function $f \in \mathcal{F}$ with the unknown distribution $p(\overrightarrow{x}, y)$ so as to decide whether it "reflects" or "approximates" it, and eventually decide that the "best" function is f^*.

1.2.1 Loss Functions and Risk

To measure how good a candidate function f is, we need some sort of quantification of the error we are having when using f as a decision function. This is where the notion of *loss function* comes at hand:

Definition 1.1 (Loss Function)**.** Given a decision function f, a *loss function* is a function $L : \mathcal{X} \times \mathcal{Y} \times \mathcal{D} \to [0, \infty)$ that measures the loss incurred by a triplet $(\overrightarrow{x}, y, f(\overrightarrow{x}))$, with the property that $L(\overrightarrow{x}, y, y) = 0 \ \ \forall \overrightarrow{x} \in \mathcal{X}, y \in \mathcal{Y}$.

Note that the minimum of a loss function L is 0, which is attained whenever the prediction for a pattern is exactly the label associated to that pattern. The loss function is thus useful only when we know which is the target y corresponding to pattern $\overrightarrow{x}$, so it cannot be applied in principle to unseen samples.

Although L is allowed to depend explicitly on $\overrightarrow{x}$, most of the loss functions used in practice do not depend explicitly on the patterns themselves, and just evaluate the difference between y and $f(\overrightarrow{x})$. Next, we introduce some of the most common loss functions. We begin with the binary classification case. The loss function

$$L(\overrightarrow{x}, y, f(\overrightarrow{x})) = \begin{cases} 0 & \text{if } \ y = f(\overrightarrow{x}) \\ 1 & \text{otherwise} \end{cases} \tag{1.2}$$

is called *0–1 loss* for obvious reasons. It is used for counting the number of misclassifications, since every time we classify wrongly it gives a value of 1. Note that this loss does not distinguish between the two kinds of misclassifications we can incur in: *false positives* when we predict $+1$ when it should be -1, and *false negatives* when we predict -1 instead of $+1$. Therefore, it is mainly used in the particular case where the set of decisions $\mathcal{D}$ is equal to the output space $\mathcal{Y} = \{+1, -1\}$.

However, in some cases it may be necessary to distinguish between misclassifications: as a simple example, consider the problem of deciding whether a patient has a tumor in the body or not. Obviously, making a mistake by saying that the patient has a tumor when this is false (that is, a false positive) is much less critical than overlooking a tumor (a false negative). This is the reason why usually the loss function above is weighted so as to reflect this difference in the errors.

Assuming now that $\mathcal{D} = \mathbb{R}$, while keeping $\mathcal{Y} = \{+1, -1\}$, the decision function f gives a real value that indicates the confidence of the prediction (actually, the confidence is given by $|f(\overrightarrow{x})|$). The most used loss function for this case is the so-called *soft-margin classification p-loss* [9], which is defined as

$$L(\overrightarrow{x}, y, f(\overrightarrow{x})) = \max\left\{0, (1 - yf(\overrightarrow{x}))^p\right\} = \begin{cases} 0 \ \text{ if } \ yf(\overrightarrow{x}) \geq 1 \\ (1 - yf(\overrightarrow{x}))^p \ \ \text{otherwise.} \end{cases} \tag{1.3}$$

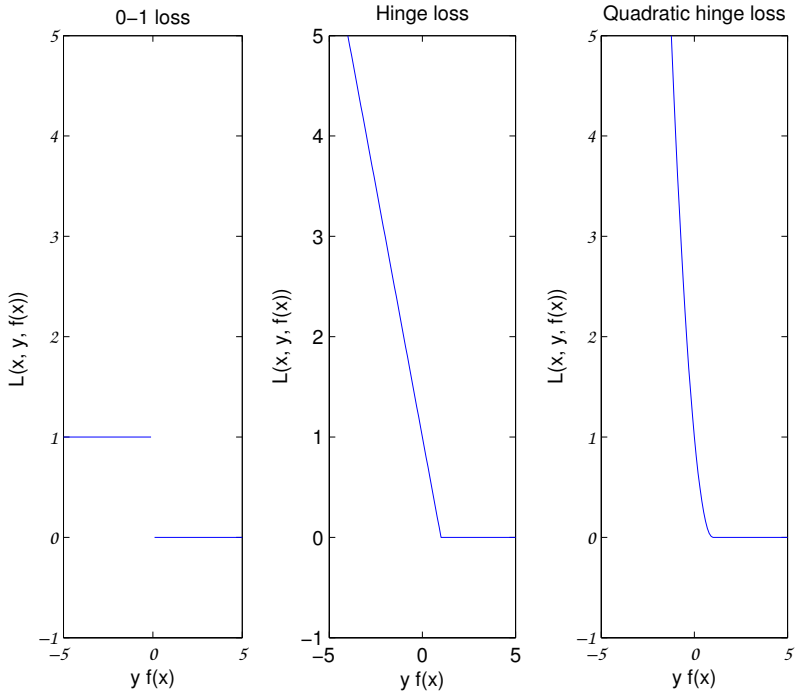

FIGURE 1.1: Different loss functions for binary classification

Note that this loss assesses the quality of the prediction by examining the product $yf(\overrightarrow{x})$. It only assumes correct classification when this value is at least 1. When this is not so, the loss penalizes the distance to 1, raised to the power p. Usually, p is set to 1 or 2. When $p = 1$, we usually call it *hinge loss*, whereas for $p = 2$ we refer to *squared (hinge) loss*. These two cases, as well as the 0–1 loss (which can also be considered as this loss when $p = 0$), are depicted for comparison in Figure 1.1.

In the case of regression we assume $\mathcal{D} = \mathbb{R} = \mathcal{Y}$. Normally, instead of inspecting the product $yf(\overrightarrow{x})$, we check the difference $y - f(\overrightarrow{x})$. This gives rise to the so–called *regression 1–loss* and *regression 2–loss* (the latter is commonly called *squared loss*, which should not be confused with the squared hinge loss), defined respectively as

$$L(\overrightarrow{x}, y, f(\overrightarrow{x})) = |y - f(\overrightarrow{x})| \tag{1.4}$$

$$L(\overrightarrow{x}, y, f(\overrightarrow{x})) = (y - f(\overrightarrow{x}))^2. \tag{1.5}$$

Similar in spirit to the soft–margin loss, another common choice is the ϵ*–insensitive* p*–loss* [10], whose parameter ϵ defines a region where the prediction is considered accurate enough so as not to penalize it:

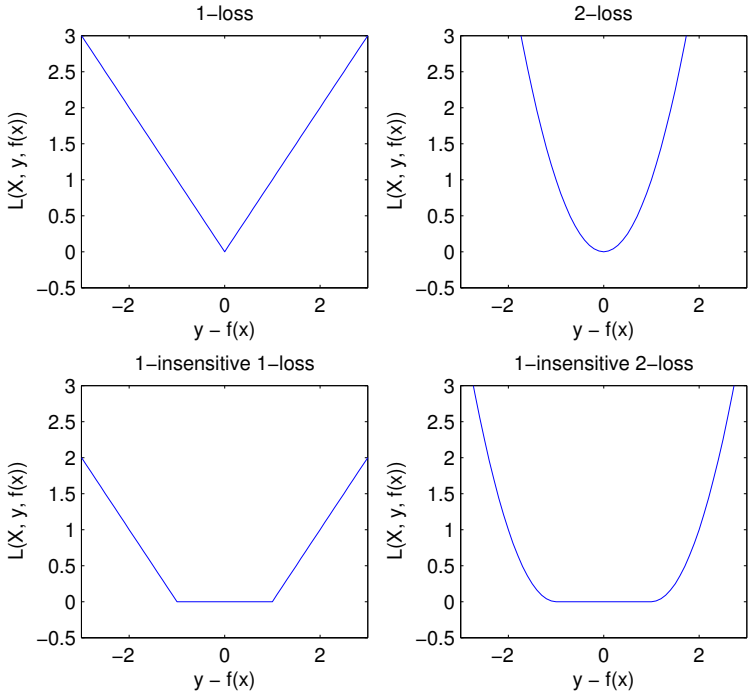

FIGURE 1.2: Different loss functions for regression

$$L\left(\overrightarrow{x}, y, f\left(\overrightarrow{x}\right)\right) = \max\left\{0, \left(\left|y - f\left(\overrightarrow{x}\right)\right| - \epsilon\right)^p\right\} = \begin{cases} 0 \text{ if } \left|y - f\left(\overrightarrow{x}\right)\right| \leq \epsilon \\ \left(\left|y - f\left(\overrightarrow{x}\right)\right| - \epsilon\right)^p \text{ otherwise.} \end{cases} \tag{1.6}$$

Outside this region, the difference is penalized with the power p. Again, the usual choices for p are 1 and 2. If $\epsilon = 0$, this loss reduces respectively to the 1–loss and the squared loss. Figure 1.2 depicts these cases, assuming an $\epsilon = 1$.

Loss functions will be used to assess the quality of a decision function f. In the particular case when test patterns are available in the training phase, we may seek to minimize the error on that very test set

$$\mathcal{T}_{test} = \left\{\overrightarrow{x}'_1, \ldots, \overrightarrow{x}'_m\right\} \in \mathcal{X}^m,$$

which has m test patterns, also assumed to be drawn i.i.d. (the $'$ superscripts are used to emphasize that these patterns are different from the training ones). Note that they, in principle, do not have labels. This error is defined in the following way.

Definition 1.2 (Test Error)**.** Given the conditional distribution $p\left(y|\overrightarrow{x}\right)$ of the data and a loss function L over a decision function f, the *test error* is the expected error of f on the test set, and is defined with the formula

$$R_{test}\left[f\right] = \frac{1}{m}\sum_{i=1}^{m}\int_{\mathcal{Y}} L\left(\overrightarrow{x}'_i, y, f\left(\overrightarrow{x}'_i\right)\right) dp\left(y|\overrightarrow{x}'_i\right),$$

where $dp\left(y|\overrightarrow{x}'_i\right)$ denotes the integral with respect to the conditional distribution.

Note that the above definition is dependent on the test set, so strictly speaking we should write $R_{test}\left[f|\tau_{test}\right]$, but we keep the current notation for the sake of simplicity.

In such a case, we could think of finding $f^* = argmin_{f\in\mathcal{F}}\left\{R_{test}\left[f\right]\right\}$. However, this minimization is far from trivial (recall that the underlying distribution p is unknown), and not very useful in practice. This is mainly due to two reasons: 1) usually the test patterns are not available during the training phase, 2) even if they are, what we would like to do is to generalize well, that is, perform well for (ideally) any possible test set, not only for a specific one [10].

Thus, it is natural to minimize the expectation of the training error over all possible training sets. This is what is known as *expected risk*.

Definition 1.3 (Expected Risk)**.** The *expected risk* (also *Bayesian risk* and *generalization error*) is the expected error over all possible test patterns, which is given by

$$R\left[f\right] = E\left[R_{test}\left[f\right]\right] = \int_{\mathcal{X}\times\mathcal{Y}} L\left(\overrightarrow{x}, y, f\left(\overrightarrow{x}\right)\right) p\left(\overrightarrow{x}, y\right) d\overrightarrow{x}\, dy,$$

where the expectation is with respect to all test patterns.

It seems that we did not progress much, since minimizing the expected risk is generally also intractable, because we do not know the distribution $p\left(\overrightarrow{x}, y\right)$. If we knew it, or had a reasonable estimate $\widehat{p}\left(\overrightarrow{x}, y\right)$ for it, we could apply *Bayesian theory* and *Bayesian strategies* to minimize the expected risk. Nevertheless, in this thesis we will not cover Bayesian techniques and instead will consider a more straightforward approach.

This approach consists of the following main idea: estimating the distribution of the data by simple analysis of them. We cannot rely on a test set for this estimation, since possibly there will not be any test patterns. All we have at our disposal is the training set $\mathcal{T}$. This set defines an empirical distribution

$$p_{emp}\left(\overrightarrow{x}, y\right) = \frac{1}{N}\sum_{i=1}^{N}\delta\left(\overrightarrow{x}_i, y_i\right),$$

where $\delta_{\overrightarrow{x}_i, y_i}$ denotes the Dirac peak at $(\overrightarrow{x}_i, y_i)$. Substituting this empirical distribution on the definition of the expected risk leads to the *empirical risk*:

Definition 1.4 (Empirical Risk)**.** The *empirical risk* (also *training error*) is an estimation of the expected risk of the form

$$R_{emp}\left[f\right] = \int_{\mathcal{X}\times\mathcal{Y}} L\left(\overrightarrow{x}, y, f\left(\overrightarrow{x}\right)\right) p_{emp}\left(\overrightarrow{x}, y\right) dxdy = \frac{1}{N}\sum_{i=1}^{N} L\left(\overrightarrow{x}_i, y_i, f\left(\overrightarrow{x}_i\right)\right).$$

Note that we have solved the two problems mentioned above: there is no need for a test set and we are estimating a general quantity, which is now computationally tractable (and in fact very easy to compute).

Does this mean that all we have to do is finding f^* as $argmin_{f\in\mathcal{F}}\left\{R_{emp}\left[f\right]\right\}$? This is what is known as *Empirical Risk Minimization* (ERM), but the answer is no. We will see in the next sections how we need to limit the function space $\mathcal{F}$ so as to obtain machines that generalize properly.

1.2.2 Overfitting and Consistency

To begin with, let us see intuitively why the function space should not be too complex. Imagine a training set of five points distributed as in Figure 1.3. The figure also shows two possible functions that explain the data.

The one in blue explains them "perfectly" in the sense that it passes through all the data instances, but its complexity seems too high for the problem at hand. It does not seem very likely that it can explain unseen points properly (for example, points lying between the samples). This is what is known in Machine Learning as *overfitting*.

Another possible explanation is the green line, which has errors for all the points (that is, it does not pass through any of them), but is probably a too simplistic model for the data, not generalizing properly either. This is the opposite phenomenon, known as *underfitting*.

Thus, the idea is to find a balance between simplicity and complexity that gives at the same time good generalization ability and does not lead to overly complex decision functions. If we just minimize the empirical risk, we will most likely end up with

overfitting (observe that the error of the blue model is exactly 0), so we need some instrument to penalize complex models and allow for some errors. However, we should take care not to penalize them too much, as that would drive to underfitting.

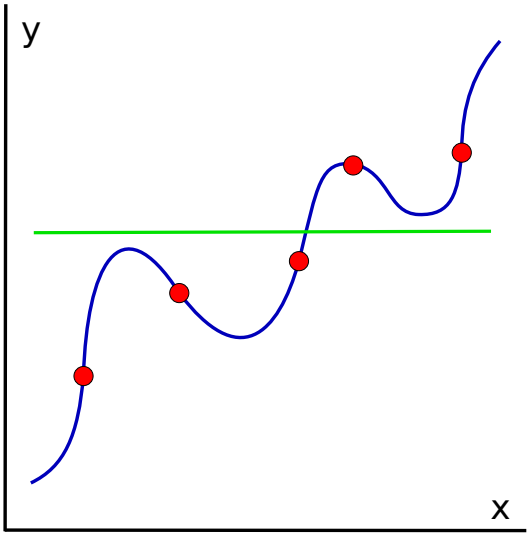

FIGURE 1.3: Example of the phenomena of overfitting (blue) and underfitting (green).

One option to do this is to somehow limit the function space $\mathcal{F}$ we can take our decision functions from. If we allow $\mathcal{F}$ to be very large, we will be able to find a function with very small expected risk (or even 0), leading to overfitting. On the other hand, $\mathcal{F}$ should be expressive enough so that underfitting is avoided and f^* generalizes well enough.

Actually, what is happening is that minimizing the empirical risk in Definition 1.4 does not necessarily yield to a minimization of the expected risk in Definition 1.3. It is important to know under which conditions this does indeed happen.

Let us consider for this discussion the classification case with labels ± 1 and the 0–1 loss (1.2). Then, we can define the quantity $\xi_i = \frac{1}{2}\left|f\left(\overrightarrow{x}_i\right) - y_i\right|$, which is either 0 or 1, if we assume that $\mathcal{D} = \{+1, -1\}$. Since the examples are i.i.d., variables $\xi_i, i = 1, \ldots, N$ are drawn in Bernoulli trials from the random variable $\xi = \frac{1}{2}\left|f\left(\overrightarrow{x}\right) - y\right|$.

We are interested in knowing how the empirical mean $\frac{1}{N}\sum_{i=1}^{N}\xi_i$ converges to $E[\xi]$. Chernoff bound [11] helps for this:

Proposition 1.5 (Chernoff Bound)**.** *Let z_i, $i = 1, \ldots, N$, be N independent instances of a random variable Z. Given any $\epsilon > 0$, we have $P\left(\left|\frac{1}{N}\sum_{i=1}^{N} z_i - E[Z]\right| \geq \epsilon\right) \leq 2\exp\left(-2N\epsilon^2\right)$,*

which is generalized by the Hoeffding bound [12]:

Proposition 1.6 (Hoeffding Bound)**.** *Let z_i, $i = 1, \ldots, N$ be as in Proposition 1.5. If the values of Z are in $[a, b]$ we have $P\left(\left|\frac{1}{N}\sum_{i=1}^{N} z_i - E[Z]\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2N\epsilon^2}{(b-a)^2}\right)$.*

If we substitute $\xi = \frac{1}{2}|f(\overrightarrow{x}) - y|$ for Z in the Chernoff bound and apply Definitions 1.4 and 1.3, we get the following result for a given decision function f:

$$P\left(|R_{emp}[f] - R[f]| \geq \epsilon\right) \leq 2\exp\left(-2N\epsilon^2\right).$$

This means that the training error approaches the generalization error exponentially fast as the number of samples increases, and in the limit of infinite samples they become identical. Putting this in probabilistic terms, the training error is an *unbiased estimate* of the generalization error and the *convergence in probability* is exponentially fast. We will write this shortly as $|R_{emp}[f] - R[f]| \to_p 0$ as $N \to \infty$. This can be generalized to the rest of loss functions in the previous section by means of the Hoeffding bound, but we omit the details.

Thus, at first sight, it seems that ERM should work, because the training error gets arbitrarily close to the expected error (provided that we have an infinite source of patterns). However, we must be rigorous: this result does not imply that minimizing the empirical risk automatically leads to minimizing the expected risk. This is illustrated in Figure 1.4.

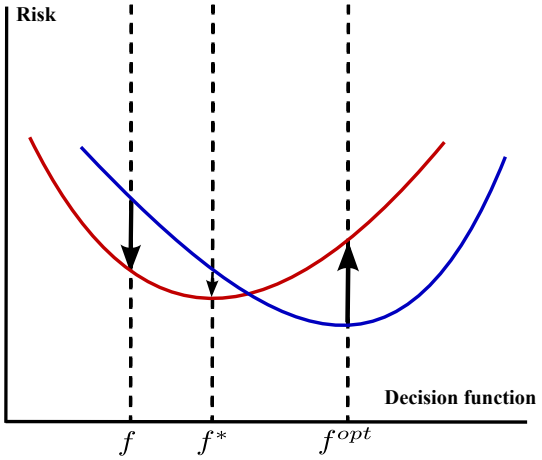

FIGURE 1.4: Illustration of why minimizing the empirical risk (in red) does not imply minimization of the expected risk (in blue).

Assume for the sake of visualization that the function space $\mathcal{F}$ is unidimensional. By applying ERM we find the function f^*, where the minimum of the empirical risk is attained. Unfortunately, we would like to obtain f^{opt}, because this is the function that minimizes the expected risk and thus performs best in test. Even if the empirical risk

converges in probability to the expected risk, this does not imply that minimizing the empirical risk is equivalent to minimizing the expected risk.

This does not hold because the function obtained by ERM is dependent on the sample. For this convergence to happen, there is the further requirement of *consistency*, which basically means that the training and the test error should be asymptotically equal for any $f \in \mathcal{F}$.

Let us use the symbol f_N to explicitly denote the dependence of f on the training set of size n (opposite to f^{opt}, which is unique). We have the following:

$$\begin{aligned} 0 &\leq R\left[f_N^*\right] - R\left[f^{opt}\right] + R_{emp}\left[f^{opt}\right] - R_{emp}\left[f_N^*\right] \\ &= R\left[f_N^*\right] - R_{emp}\left[f_N^*\right] + R_{emp}\left[f^{opt}\right] - R\left[f^{opt}\right] \\ &\leq \sup_{f\in\mathcal{F}} \left\{R\left[f\right] - R_{emp}\left[f\right]\right\} + R_{emp}\left[f^{opt}\right] - R\left[f^{opt}\right], \end{aligned}$$

where the first inequality holds because, by definition, $R\left[f^{opt}\right] \leq R\left[f_N\right]\ \forall f_N \in \mathcal{F}$ and $R_{emp}\left[f_N^*\right] \leq R_{emp}\left[f_N\right]\ \forall f_N \in \mathcal{F}$.

In the last inequality, we know that the second half of the right-hand side converges to 0 as $N \to \infty$. The problem is that in the second line, $R\left[f_N^*\right] - R_{emp}\left[f_N^*\right]$ does not necessarily tend to 0, because f_N^* is dependent on the training set. If the first half also tends to 0, we say that there is *uniform convergence of risk*.

Definition 1.7 (Uniform Convergence of Risk)**.** Given a function space $\mathcal{F}$, the risk is said to *converge uniformly* if $\sup_{f\in\mathcal{F}} \left\{R\left[f\right] - R_{emp}\left[f\right]\right\} \to 0$ as $N \to \infty$.

Here the dependence on N lies in the calculation of the empirical risk of Definition 1.4 over training sets of that size. Thus, we see that uniform convergence of the risk is a necessary condition for ERM to converge to expected risk minimization. It turns out that this condition is also sufficient, as stated in the following theorem [13].

Theorem 1.8. *Uniform convergence of the risk*

$$\lim_{N\to\infty} P\left(\sup_{f\in\mathcal{F}} \left\{R\left[f\right] - R_{emp}\left[f\right]\right\} > \epsilon\right) = 0$$

$\forall \epsilon > 0$ *is a necessary and sufficient condition for ERM to converge to expected risk minimization.*

At this point, we are ready to formulate the following fact: if we want to design learning machines that generalize well, we should limit the space of possible decision functions $\mathcal{F}$ so that uniform convergence of the risk is ensured. The next section will explain how this can be achieved.

1.2.3 Structural Risk Minimization

We have seen how the ERM principle is not enough if we want to generalize well. Minimizing the empirical risk will tend to give models too tailored to the specific training set, incurring in overfitting. If we somehow penalize the complexity of the models obtained, we will be able to get simpler models that are more plausible, explanatory and reliable than complex ones. This is usually referred as compliance with Occam's Razor: *entia non sunt multiplicanda praeter necessitatem* ("entities should not be multiplied beyond necessity").

However, we have to keep in mind that simplifying the models too much can lead to underfitting. Therefore, we need basically two things: 1) an accurate measure of the underlying complexity of some model, and 2) a framework that takes into account this complexity to give convenient models.

Regarding these two requirements, Vapnik and Chervonenkis developed the so–called *Structural Risk Minimization* (SRM) principle [10], based on a measure of complexity called the *VC dimension*. Here, VC stands for Vapnik–Chervonenkis. In the next paragraphs, we briefly explain them.

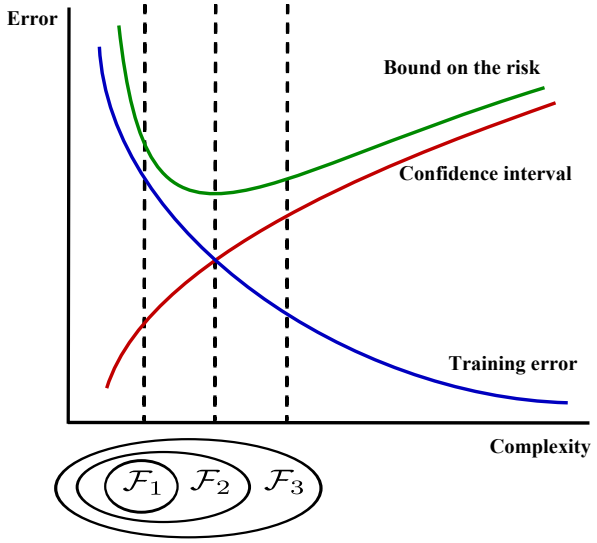

FIGURE 1.5: Scheme of Structural Risk Minimization. In this case, the subset chosen would be $\mathcal{F}_2$, since it gives the lowest value for the risk bound.

SRM is based on the following idea: divide the function space $\mathcal{F}$ in subsets $\mathcal{F}_1, \mathcal{F}_2, \ldots$ of increasing complexity, and pick the one that has the best balance between complexity and training error. This is illustrated in Figure 1.5, where we see that, the higher the complexity, the lower the training error. On the other hand, the higher the complexity, the higher the model's confidence interval. As we will see later in detail, we can derive a bound on the risk that depends on both terms. The subset which is finally chosen is the one that provides the best trade–off between these two conflicting terms.

In order to understand the concept of VC dimension, we need first to introduce the idea of *shattering*.

Definition 1.9 (Shattering)**.** Given a family C of sets and a set A, C is said to *shatter* A if, for every subset S of A, there is some $U \in C$ such that $A \cap U = S$.

That is to say, C shatters A if C is more expressive than A in the sense that, whichever subset we take from A, we can also take an element of C that includes this subset. In our case, we assume A to be a finite set. Specifically, A will be a training set $\mathcal{T}$ like the one in (1.1). We will further assume to be in the binary classification case for the sake of simplicity (all these concepts can be generalized to the regression case, but the notation becomes quite cumbersome [3]).

The notion of VC dimension directly depends on the so-called *shattering coefficient*:

Definition 1.10 (Shattering Coefficient)**.** Given a family C of sets, we define the *N-th shattering coefficient* of C as

$$\mathcal{S}_C(N) = \max_{\mathcal{T} \in (\mathcal{X} \times \mathcal{Y})^N} \{ | \{ B \cap U : B \in \mathcal{P}(\mathcal{T}), U \in C \} | \}.$$

In the formula above, $\mathcal{P}(\mathcal{T})$ stands for the *power set* of the training set $\mathcal{T}$, that is, the set formed by all subsets of $\mathcal{T}$. This means that the N–th shattering coefficient is equal to the largest number of subsets of any training set $\mathcal{T}$ of N points that can be subsets of sets belonging to the family C. Obviously, we have $\mathcal{S}_C(N) \leq 2^N$, since $y_i = \pm 1 \ \forall i$. In the particular case where $\mathcal{S}_C(N) = 2^N$, it means that there is a training set of size N that can be shattered by C, according to Definition 1.9.

At this point, we can define the VC dimension as

Definition 1.11 (VC Dimension)**.** Given a family C of sets, the *VC dimension* of C is defined as $VC(C) = \max_N \{ N : \mathcal{S}_C(N) = 2^N \}$.

Thus, the VC dimension is the largest N such that any subset of size N can be shattered by C. If there is not such an N, $VC(C) = \infty$. In our context of Pattern Recognition,

the family C is the function space $\mathcal{F}$. We are interested in limiting $\mathcal{F}$ so that $VC(\mathcal{F})$ is not too high, since if it is very high it means that $\mathcal{F}$ is probably an overly expressive class of functions, and the danger of incurring in overfitting is high.

As a simple example, consider that $N = 3$ and that $\mathcal{F}$ is the set of separating hyperplanes in two dimensions, so that $\mathcal{X} = \mathbb{R}^2$. The output space is still considered to be ± 1: the decision function will give $+1$ at the positive side of the hyperplane, and -1 at the negative side. No matter how we label these 3 points, there is always a hyperplane that can separate them (assuming of course that the points are in general position, thus not collinear), as can be seen in Figure 1.6. If we change to 4 points, this is no longer possible in some cases, like the one in the bottom. Hence, the VC dimension of bidimensional classifying hyperplanes is 3.

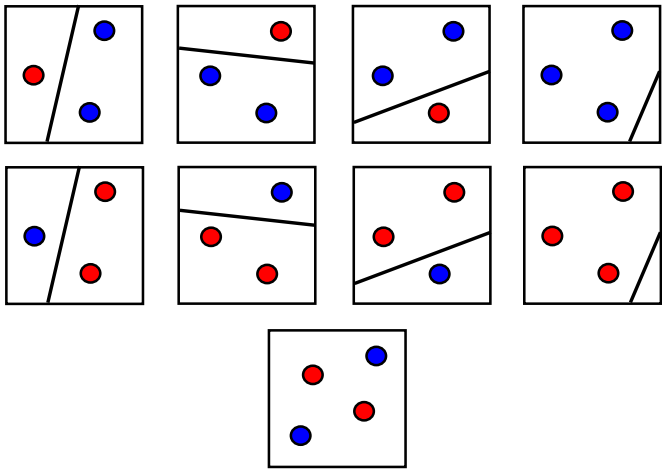

FIGURE 1.6: Example of VC dimension. The set of bilinear hyperplanes has a VC dimension of 3, because no matter how 3 patterns are labeled, they can be separated by a hyperplane. However, this is not always possible for 4 points, as in the example in the bottom.

Turning back to the general case of a sample of N points, if $VC(\mathcal{F}) \geq N$ it is possible that we can obtain a training error of 0, and thus very likely suffer from overfitting. In contrast, if $VC(\mathcal{F}) < N$ the chances of obtaining no error decay very quickly. Specifically, it can be shown [10] that

$$P(R_{emp}[f] = 0) \leq \frac{1}{2^N} \left(\frac{eN}{VC(\mathcal{F})} \right)^{VC(\mathcal{F})},$$

where the probability is with respect to the uniform distribution on the labels. We see then the confirmation of what we previously stated informally: if we keep the complexity (i.e. the VC dimension) small, it is unlikely that we obtain a model with overfitting, whereas if the complexity is too high the probability of this happening rapidly increases.

Looking again at Figure 1.5, we can quantify mathematically the term of confidence interval. If $VC(\mathcal{F}) < N$ then, independently of the underlying distribution p generating the data, for all $f \in \mathcal{F}$ and with a probability of $1 - \delta$ over the drawing of the training set, it holds that [3]

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{1}{N}\left(VC(\mathcal{F})\left(\log\frac{2N}{VC(\mathcal{F})} + 1\right) + \log\frac{4}{\delta}\right)}. \tag{1.7}$$

Note that the term of confidence interval, as depicted in the figure, increases monotonically with $VC(\mathcal{F})$, so the bound shows that the expected and empirical risks might be very different from one another if the model complexity is high.

1.2.4 Regularization

Once we know how to characterize and measure the complexity of the function space $\mathcal{F}$, we can think of two different strategies to comply with the SRM principle. The first one is to choose $\mathcal{F}$ in a way such that the complexity is limited. In our context, this would mean choosing functions with limited VC dimension.

The second option is to add a penalty term to the model complexity, so that we try to minimize not the empirical risk, but the *regularized risk*

$$R_{reg}[f] = R_{emp}[f] + C\Omega[f], \tag{1.8}$$

where $\Omega[f]$ is a measure of the complexity of model f and $C > 0$ is a trade–off parameter between the minimization of the pure empirical error and the model complexity. This technique was introduced in [14].

In practice, in SRM it is usually sought that $\mathcal{F}$ be a compact set (we will see in Chapter 2 what this means) and that $R_{emp}[f]$ be continuous on $\mathcal{F}$. The reason for this is that, taking into account the Operator Inversion Lemma [15], the map that gives the minimal empirical error (that is, the one that performs ERM) is an invertible map, so that, given the minimal empirical error we can obtain a decision function which yields it. In common language, the optimization problem is *well–posed*, something which is not true in general, since ERM is known to lead to *ill–posed* problems in the general case [16].

Nevertheless, if we did not use regularization, the problem of ERM, although well–posed, would be a constrained one, since $\mathcal{F}$ would be restricted to be compact. This is undesirable because constrained optimization problems are more difficult to deal with than unconstrained ones.

By using regularization, we can forget about the constraints on $\mathcal{F}$ if we impose some other requirements. For example, if $\Omega[f]$ and $R_{emp}[f]$ are both convex functions, the minimum of $R_{reg}[f]$ exists and is unique (cf. Chapter 2).

This is the reason why some discontinuous loss functions, such as the 0–1 Loss (1.2), are seldom convenient in Machine Learning (in fact, such choices have been shown to lead to NP–hard problems [17]). Instead, we use in practice continuous approximations, such as the hinge loss in (1.3).

1.3 Linear and Kernel Methods

In the previous section we have discussed why it is convenient to restrict the function space $\mathcal{F}$, so as to obtain good generalization properties. It seems natural that special attention has been given to the particular case of linear functions, which are the best understood in fields such as statistics, and which are also easily applicable.

In this section we review linear classifiers and regressors (generally termed as *linear machines*), which provide a powerful framework for the construction of more complex systems. We also introduce the concept of *margin*, which is of particular importance for generalization. Among these extensions, based on previous observations about ERM and SRM, we find some of the most popular paradigms in Machine Learning, such as *Neural Networks* and the so–called *Kernel Methods*.

This thesis will focus on the latter. As their name makes clear, the notion of a *kernel* is vital to understand them. After this brief review of linear algorithms, we will introduce intuitively the concept of kernel. A more detailed and formal study of kernels is postponed till Chapter 2, along with additional mathematical background required to understand the rest of the thesis.

1.3.1 Linear and Max–Margin Methods

Let us begin with the binary classification case. From now on, it is always assumed that the input space is $\mathcal{X} = \mathbb{R}^d$ or a subset of $\mathbb{R}^d$. Thus, the different features are scalars and the patterns are d–dimensional, so we will be interested in learning a decision function $f : \mathbb{R}^d \to \mathbb{R}$. Recall that some discontinuous or discrete loss functions are cumbersome to deal with, so it is natural that $\mathcal{D} = \mathbb{R}$, even if $\mathcal{Y} = \{+1, -1\}$.

Linear classifiers amount to find a decision function of the form

$$f(\overrightarrow{x}) = \langle \overrightarrow{w}, \overrightarrow{x} \rangle + b = \sum_{i=1}^{d} w_i x_i + b, \tag{1.9}$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the parameters to be learnt, and the $\langle \cdot, \cdot \rangle$ operator denotes standard *scalar* or *dot product*. In order to make the decision itself, for classification we take the sign of $f(\overrightarrow{x})$. By convention, if $f(\overrightarrow{x}) = 0$, its sign is considered to be $+1$.

The geometric interpretation of such a classifier is shown in Figure 1.7, for the simple case of $d = 2$. The decision function defines a *hyperplane* $\langle \overrightarrow{w}, \overrightarrow{x} \rangle + b = 0$ that splits the d–dimensional space in two halfspaces, each one corresponding to one of the two possible classes. This hyperplane is sometimes called *decision boundary*. As in virtually all the literature, we will refer to $\overrightarrow{w}$ as the *weight vector* and to b as the *bias*.

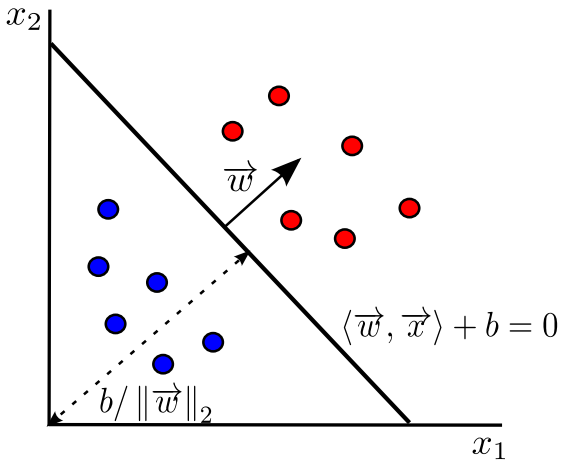

FIGURE 1.7: Geometric interpretation of a linear classifier. It defines a hyperplane that separates the positive class (in red) from the negative one (in blue).

The weight vector $\overrightarrow{w}$ defines a direction perpendicular to the hyperplane and pointing to the positive class. The bias b indicates the perpendicular distance from the hyperplane to the origin $\overrightarrow{0}$. Note that if $b = 0$ the hyperplane $\langle \overrightarrow{w}, \overrightarrow{x} \rangle = 0$ passes through $\overrightarrow{x} = \overrightarrow{0}$.

This very simple classifier is commonly known among statisticians as *linear discriminant* [18]. In the Neural Network community, it is called *perceptron*, and the number $-b$ is usually called *threshold* and denoted by θ [19].

The first iterative algorithm to learn such a linear classifier was the "δ–rule" designed by Frank Rosenblatt in the 50s [20]. However, the are two big problems with such an approach. The first one is that the solution depends on the order in which the patterns are presented (the algorithm learns on an online basis, that is, pattern by pattern). The second one is that, provided that the classes are *linearly separable* (that is, they can be separated by a hyperplane), there are infinitely many possible separating hyperplanes.

According to what was discussed in previous sections, we are especially interested in obtaining a single global minimum for the risk. In this framework of linear classifiers, this means finding the hyperplane that minimizes the risk. Here is where the notion of *margin of a pattern*, *margin of a training set* and *maximal margin hyperplane* come at hand.

Definition 1.12 (Margin of a Pattern)**.** The *margin of a pattern* $(\overrightarrow{x}_i, y_i)$ with respect to a hyperplane $\langle \overrightarrow{w}, \overrightarrow{x} \rangle + b = 0$ is the quantity

$$M\left(\overrightarrow{x}_i, y_i; \overrightarrow{w}, b\right) = y_i \left(\langle \overrightarrow{w}, \overrightarrow{x}_i \rangle + b\right).$$

In the particular case where the hyperplane is normalized to $\langle \overrightarrow{w} / \left\| \overrightarrow{w} \right\|_2, \overrightarrow{x} \rangle + b / \left\| \overrightarrow{w} \right\|_2$, it is called *geometric margin of a pattern*.

The geometric margin of a pattern can be seen to be the perpendicular distance from that pattern to the classification hyperplane. If we consider the full training set, we have

Definition 1.13 (Margin of a training set)**.** The *(geometric) margin of a training set* $\mathcal{T}$ is defined as the minimal (geometric) margin of its patterns, i.e.

$$M_{\mathcal{T}}\left(\overrightarrow{w}, b\right) = \min_{(\overrightarrow{x}_i, y_i) \in \mathcal{T}} \left\{ M\left(\overrightarrow{x}_i, y_i; \overrightarrow{w}, b\right) \right\}.$$

These two definitions are illustrated in Figure 1.8, assuming a normalized hyperplane. Three points are highlighted: the first two belonging to the positive class, and the third one to the negative class. Considering the hyperplane $(\overrightarrow{w}, b)$ selected, the margin of the training set is $M_{\mathcal{T}}\left(\overrightarrow{w}, b\right) = M\left(\overrightarrow{x}_1, y_1; \overrightarrow{w}, b\right)$, since the first point is the one closest to it.

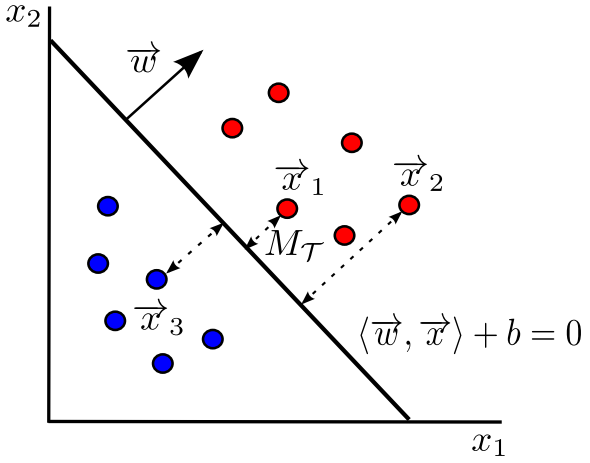

FIGURE 1.8: Illustration of the geometric margin of three points and the geometric margin of the training dataset, given by the first point.

At this point, it should be anticipated what do we mean by *maximal margin hyperplane*:

Definition 1.14 (Maximal Margin Hyperplane)**.** Given a training set $\mathcal{T}$, the *maximal margin hyperplane* is the hyperplane $\langle \overrightarrow{w}^*, \overrightarrow{x} \rangle + b^* = 0$ that yields the maximal margin of $\mathcal{T}$, which is found by solving the problem

$$(\overrightarrow{w}^*, b^*) = \underset{\overrightarrow{w} \in \mathbb{R}^d, b \in \mathbb{R}}{\arg\max} \left\{ M_{\mathcal{T}} \left(\overrightarrow{w}, b \right) \right\}.$$

Therefore, the maximal margin hyperplane, informally, is the one that separates the most the classes. This hyperplane can be shown to be unique up to scaling [2], so apparently we have solved the problem of having infinitely many separating hyperplanes, provided that we can find it in an effective way.

Furthermore, intuitively it should also be the one that generalizes best, since it is the one furthest from the samples of both classes. It should maximize the probability of classifying correctly unseen points; if it were very close to one of the classes, for instance, a small variation in a point from that class might very well position it in the wrong side of the hyperplane. We will confirm this intuition in Chapter 3 by means of the SRM principles outlined previously.

For now, let us examine briefly what happens when the classes are linearly inseparable. In that case, the perceptron algorithm with δ–rule never converges, so it fails to find a solution. In fact, there is no possible solution, unless we are more flexible and allow for misclassifications. To this aim, we define the following quantity:

Definition 1.15 (Margin Slack Variable)**.** Given a fixed value $\gamma > 0$, the *margin slack variable* of a pattern $(\overrightarrow{x}_i, y_i)$ w.r.t. the hyperplane $\langle \overrightarrow{w}, \overrightarrow{x} \rangle + b = 0$ is the quantity $\xi_i = \max \{0, \gamma - y_i \left(\langle \overrightarrow{w}, \overrightarrow{x}_i \rangle + b \right)\}$.

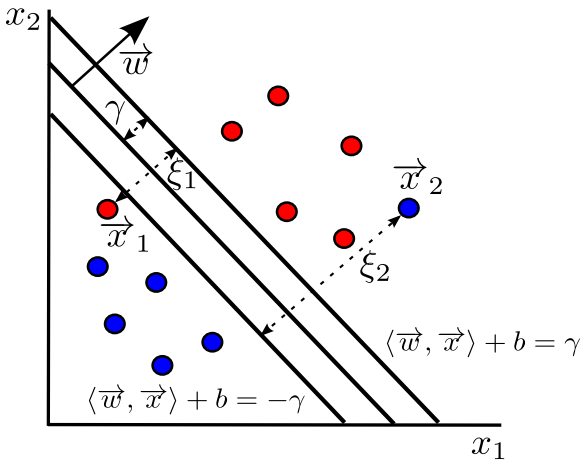

FIGURE 1.9: Illustration of margin slack variables. The classes are no longer linearly separable, so we allow for misclassifications ξ_i with a tolerance γ.

Thus, ξ_i measures how far a point is from the correct side of the hyperplane, with a "tolerance" γ. If a point is misclassified, we have $\xi_i > \gamma$, since $y_i \left(\langle \overrightarrow{w}, \overrightarrow{x}_i \rangle + b \right) < 0$. If the point is classified correctly, but its margin is less than γ, we get $0 < \xi_i < \gamma$. Figure 1.9 depicts a case with two misclassified points and linearly inseparable classes.

Moving to the case of regression, the decision function is still of the form (1.9). The idea is to find the hyperplane that "best" fits the data. Since in practice the points will not be collinear, again we have to allow for errors. Consider for instance the slack variables $\xi_i = |y_i - (\langle \overrightarrow{w}, \overrightarrow{x}_i \rangle + b)|$.

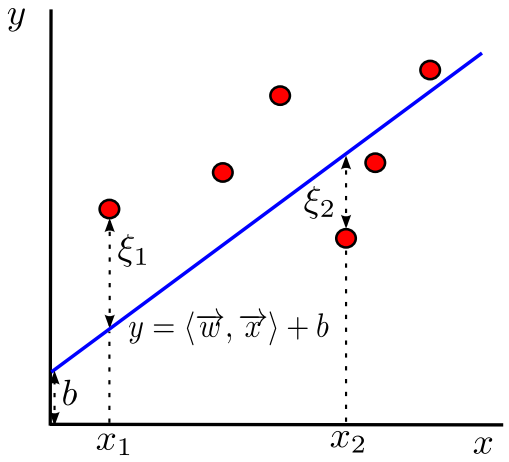

FIGURE 1.10: Illustration of Regression Slack Variables. Whenever we do not have collinear points, we allow for errors in regression quantified by slacks ξ_i.

Figure 1.10 shows such a case, where for plotting purposes we consider $d = 1$, so that the inputs are just scalars. The bias b now expresses the distance from the hyperplane to the origin, since $f(0) = b$. The slacks ξ_i will be positive whenever the points do not lie exactly on the regression hyperplane.

An attentive reader might have realised that these slack variables, both in classification and regression, are very related to the loss functions explained in previous sections. For example, in classification the hinge loss (1.3) corresponds to margin slack variables in Definition 1.15 with a tolerance of $\gamma = 1$, whereas the slacks mentioned in regression are associated to regression 1–loss (1.4). This connection will become evident in Chapter 3, where we will see how this slack quantification is used in a regularized risk framework.

1.3.2 A Note on Kernel Functions

For the moment, let us change slightly the topic. Recall from the previous section that the decision function learnt by linear machines is of the form $f(\overrightarrow{x}) = \langle \overrightarrow{w}, \overrightarrow{x} \rangle + b$.

Moreover, the slacks are calculated in such a way that some difference between the desired output (e.g. γ or y) and $f\left(\overrightarrow{x}\right)$ is evaluated.

This means that the only operations performed between inputs $\overrightarrow{x}$ and weight vectors $\overrightarrow{w}$ are dot products. If we were able to generalize somehow this dot product, we could effectively build more powerful machines. What do we refer to when talking about generalizing the dot product? For instance, it would be very convenient to be able to evaluate a dot product in a space with higher dimensionality than the d–dimensional input space.

If we could do so, we would be able to train linear machines in that high-dimensional space, which, translated back to the input space, would yield non–linear machines. Hence, with a linear machine framework we would be able to learn Non–Linear Machines. Perhaps the most obvious reason to do this in classification is that, even if the patterns are linearly inseparable in the input space, they might be linearly separable in the high-dimensional space.

This idea is illustrated in Figure 1.11. The map from the input space to the high-dimensional space is denoted by $\Phi\left(\cdot\right)$. It is almost a universal convention to call this high-dimensional space the *feature space* and denote it by $\mathcal{H}$, so we will follow it (the $\mathcal{H}$ comes from it being a Hilbert space, as will be seen in Chapter 2). For this reason, this map is called the *feature map*. The dimension of the feature space will be denoted by $d_{\mathcal{H}}$, and it will be seen that this dimension can even be infinite.

In the figure we can see how the feature map $\Phi\left(\cdot\right)$ makes the linearly inseparable patterns from Figure 1.9 become linearly separable. In this particular case the feature space is also taken to be bidimensional for the sake of simplicity, but this will not be normally the case. What is important to note is that, once we have learned a separating hyperplane in that space, it translates to a non-linear decision boundary in the input space, which can separate both classes.

Notice the notation, which will be our convention from now on. The linear machine is learned in the feature space, so patterns $\overrightarrow{x}_i$ are effectively substituted by $\Phi\left(\overrightarrow{x}_i\right)$. Thus, we switch from d–dimensional inputs to $d_{\mathcal{H}}$–dimensional projections of these inputs. In order to perform a dot product in the feature space, the weight vector must be also $d_{\mathcal{H}}$–dimensional and lie in the space. This is emphasized notationally with the subscript $\mathcal{H}$.

The most important fact about kernel functions is that, once we have learnt the linear machine in the feature space, we do not need the explicit knowledge of the inverse map $\Phi^{-1}\left(\cdot\right)$ to go back to the input space. What is more, we will see in the next chapter that it is not even necessary to know explicitly the feature map $\Phi\left(\cdot\right)$.

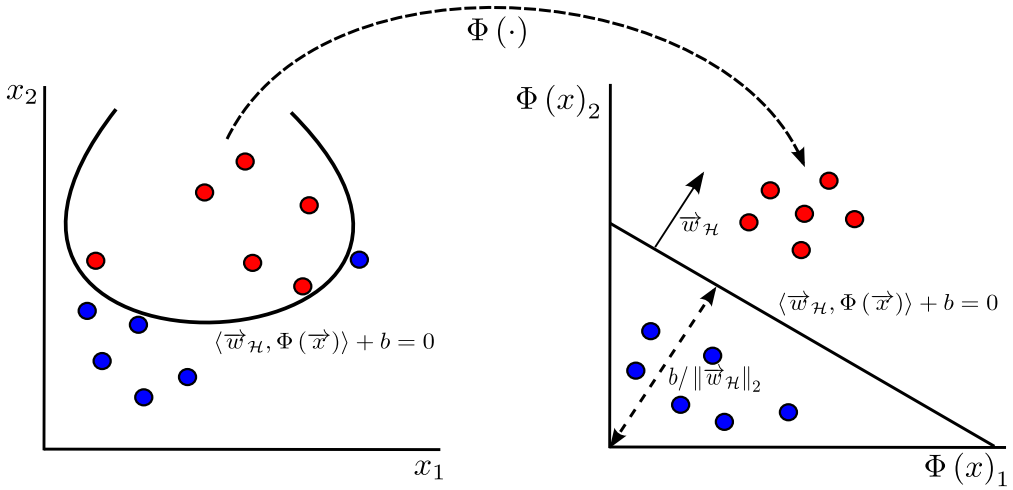

FIGURE 1.11: Projecting the patterns to a feature space can make a linear machine learnt there translate to a non–linear machine back in the input space, more powerful to deal with difficult problems.

This is so because, as pointed previously, the only operations between vectors are dot products. At this point, we can define, somewhat informally, a kernel function as follows. More details will be given in Chapter 2.

Definition 1.16 (Kernel Function)**.** Given an input space $\mathcal{X}$, a feature space $\mathcal{H}$ endowed with a dot product, and a (feature) map $\Phi : \mathcal{X} \to \mathcal{H}$, a *kernel function* (or simply a *kernel*) is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that, given any two inputs $\overrightarrow{x}, \overrightarrow{x}' \in \mathcal{X}$, it calculates the dot product of these two patterns in the feature space, that is

$$k\left(\overrightarrow{x}, \overrightarrow{x}'\right) = \left\langle \Phi\left(\overrightarrow{x}\right), \Phi\left(\overrightarrow{x}'\right) \right\rangle. \tag{1.10}$$

Therefore, by making use of kernel functions, we can effectively substitute all dot products in the input space with dot products in the feature space, and thus learn implicitly a classifier or a regressor in that space, even if we do not know the nature of the feature map $\Phi(\cdot)$ underlying in the kernel. This is what is known in the literature as the *kernel trick*, introduced in [21].

In order to make this trick useful, the kernel function should give a measure of the similarity of patterns. As suggested by Figure 1.11, ideally the function should give similar results for the points of the same class, so that they are clustered together and become linearly separable from the points of the other class. This is also the desideratum in regression, because similar points typically should result in similar outputs.

In the next chapter we will characterize mathematically the conditions a kernel function has to fulfil so that the kernel trick works. We will also explain what do we mean by "similarity" of patterns and give examples of typical kernels used in practice.

1.4 Thesis Roadmap

Once we know the fundamental ideas of Pattern Recognition and Kernel Methods, we are now ready to motivate in this section the rest of the thesis and describe briefly its structure.

1.4.1 Motivation

Support Vector Machines (SVMs) are probably the best–known variant of Kernel Methods in the Machine Learning and Pattern Recognition community. They were initially proposed by Vladimir Vapnik in the work [21]. By making use of the kernel trick and basing upon the idea of maximum margin hyperplanes, they show very good generalization abilities for classification and regression problems.

This generalization performance of course is not mere chance, but it is based very soundly on the SRM principle that had been studied previously by Vapnik himself in [10]. Maximum margin hyperplanes do have a limited VC dimension, yet they are sufficiently expressive for many of the problems encountered in practice, because the use of kernels results in non–linear machines in the input space.

Traditionally, classification with SVMs (*Support Vector Classification*, SVC) and regression (*Support Vector Regression*, SVR) have been treated separately, even if they share most of their characteristics. As another component of diversity, some other variants based on SVMs have arisen in the literature. Among these, we will focus particularly on *Least-Squares Support Vector Machines* (LS-SVMs), introduced in [22], and ν–SVMs [23].

In parallel, there exists a line of research based on the geometric interpretation of SVC and SVR, initiated with the papers [24, 25], aside from the notions of hyperplane and margin. There, it was shown that SVC is equivalent to the problem of finding the two closest points in two convex hulls, whereas SVR reduces to the same problem, once we increment the dimensionality of the patterns to cover the width of the ϵ–insensitive loss function used.

Therefore, several algorithms have been devised for SVMs based either on their original formulation or on their geometric counterparts. A milestone in the first modality was the conception of the *Sequential Minimal Optimization* (SMO) algorithm by John Platt in his work [26]. Its importance is evident if we consider that it has given rise to several related algorithms, which are termed with the collective notation of "*SMO-like*" algorithms.

As for the geometric modality, some remarkable algorithms are the Gilbert–Schlesinger–Kozinec (GSK) method [27] and the Mitchell–Dem'yanov–Malozemov (MDM) one [28]. However, most of the research is still concentrated on the original formulation and not in this geometric interpretation, even if interesting insights can be gained from it.

The first main contribution of this thesis is to provide a general framework and a viewpoint that connects most of the formulations mentioned so far. Some examples of the benefits that such a framework includes are the following:

- Formulating a general optimization problem that covers SVC, SVR, ν–SVC, ν–SVR, LS–SVC and LS–SVR.
- Understanding very intuitively what the SMO algorithm does and why it works.
- Understanding extensions for it in the form of SMO–like methods, that were developed in the literature from other points of view.
- Establishing connections between the SMO–like algorithms and geometrical algorithms. The most important of these is that the MDM algorithm is a special case of the SMO one.
- Giving new and simpler proofs of the convergence of SMO and SMO–like methods (be them geometrical or not).

This last point is the second main contribution of the thesis. The SMO algorithm was proposed by Platt without any theoretical guarantee that it always converged to the optimal solution. This topic was later covered in a series of papers, culminating in [29]. There, several SMO–like algorithms, including SMO itself, were shown to converge, either in a finite number of iterations or asymptotically to the optimal solution. This is referred to as *asymptotic convergence*.

The thesis shows how, building on this common framework, we can give a much simplified proof of the asymptotic convergence of SMO and SMO–like methods. Besides, it does not only cover SVC and SVR, but also their ν and LS variants, as well as the corresponding geometrical algorithms, because of the common formulation that encompasses all these.

1.4.2 Structure

The text has been divided in eight chapters and one appendix:

- As we have just seen, Chapter 1 aims to describe a general background, so that the reader gets familiar with the basic ideas of Machine Learning and Pattern Recognition. Then it describes a bit more in detail the case of Supervised Learning, as well as some ideas specific to it that will help in understanding why SVMs, formulated in Chapter 3, work well. It also introduces the notion of kernels, paving the way for a more thorough treatment in Chapter 2. Throughout this chapter, specific terminology of the field is also gently introduced.

- Chapter 2 gives the necessary mathematical background to fully understand the following chapters. It extends the initial study on kernels of Chapter 1. It also treats optimization theory, which is vital to understand the SVM formulations in Chapter 3, since they are formulated as optimization problems in two modalities (primal and dual). In addition, we revise some important notions of geometry and algorithmical convergence, necessary to grasp the geometrical formulations of Chapter 3 and the proofs of convergence of Chapter 6.

- Chapter 3 introduces the different SVM variants treated in the thesis, separating the classification and regression cases. It explains the primal and dual formulations of each variant, as well as the mathematical and graphical characterization of optimality with the Karush–Kuhn–Tucker (KKT) conditions. After that, it explains in a similar fashion some geometrical problems that are closely related.

- Chapter 4 describes the state–of–the–art of algorithms used to solve the SVM and geometric formulations of Chapter 3. As for the first category, it concentrates on the SMO algorithm, while regarding the second one, it explains the GSK and MDM algorithms. All these descriptions are given under a common viewpoint and its associated notation, which pave the way for the unification explained in Chapter 5.

- Chapter 5 is devoted to the first main contribution of the thesis, that is, the common formulation that allows to fuse all the different formulations of Chapter 3. The idea here is formulating a primal general enough so that it covers the primal formulations of all those variants. Its dual will consequently cover all their dual formulations. After settling this generalized problem, we will see how the SMO adaptations for different variants can also be encompassed in a corresponding generalized SMO algorithm. The connection between SMO and its counterpart MDM is also highlighted.

- Chapter 6 deals with convergence aspects for the generalized SMO algorithm described in Chapter 5. First, it shows the asymptotic convergence of the algorithm, in a much simpler way than the attempts presented in the literature so far. Moreover, the fact of being general makes that with just one proof we include the

different SMO adaptations, as well as the GSK and MDM algorithms. Finally, it justifies the *shrinking* strategy used in the SMO method.

- Chapter 7 gives a summary and discussion of the contributions and results presented in this thesis. To conclude, it suggests a list of possible topics and lines for future research that would extend the material presented.

- Chapter 8 is the translation of Chapter 7 to Spanish. This is a requirement for the "Doctor Europeus" mention.

- Appendix A gives a list of references published by the author and the way they are related to the present thesis.

Chapter 2

Mathematical Background

This chapter provides the necessary mathematical background to understand the subsequent chapters of the thesis. It begins with a discussion of convergence in Section 2.1, regarding both convergence of *sequences* themselves (§2.1.1) and *rates of convergence* (§2.1.2). This is necessary before delving into a deeper mathematical description of *kernels* than the one sketched in the previous chapter in §1.3.2. Kernels are the key concept of Kernel Methods, and hence one of the main characteristics of SVMs.

The other central characteristic of SVMs is their application of the SRM principle (also introduced in the previous chapter). This application is based on solving *optimization problems* that basically minimize two terms: a first one with guaranteed low VC dimension, and a second one acting as a regularization term.

Therefore, it is essential to understand the optimization theory behind these problems solved by SVMs. It will be seen that they fall into the category called *quadratic programming*, which in turn is a specific kind of the so–called *convex programming*, explained in §2.4.2. In order to understand the latter, first we need some notions on *convex sets* in §2.3 and *convex functions* in §2.4.1.

Other important concepts that are covered in this chapter are *convex hulls* in §2.3 and *Lagrangian duality* in §2.4.3.

The material for this chapter is adapted from the books [2], [3], [30] and [31].

2.1 Convergence

2.1.1 Sequences

We begin with the concept of sequence, which is the cornerstone in the field of convergence.

Definition 2.1 (Sequence)**.** Given a set V, a *sequence* in V is a function $f : \mathbb{N} \to V$ from the set of natural numbers to V, that is to say, an ordered list of *terms* (also *members* or *elements*). If the number of terms is finite, the sequence is said to be *finite*, otherwise it is *infinite*.

Notationally, we will denote a sequence v with superscripts as $\{v^t\}$, where $t = 0, 1, \ldots$. The terms of this sequence will thus be $v^0, v^1, v^2, \ldots$, with $v^t \in V$ for all t. Depending on how these terms evolve throughout the sequence, we can have the following case, provided that we can establish an order between different elements of the set V:

Definition 2.2 (Monotone Sequence)**.** A given sequence $\{v^t\}$ is *monotonically increasing* (*monotonically decreasing*) if every term is greater (less) than or equal to the preceding one, that is, if $v^{t+1} \geq (\leq)\ v^t$ for all t. If the inequality is strict, we say that the sequence is *strictly monotonically increasing* (*strictly monotonically decreasing*), that is, $v^{t+1} > (<)\ v^t$ for all t.

Once this is clear, the idea of subsequence is simple. Intuitively, a subsequence is a sequence that can be derived from another sequence by deleting some terms, without changing the order of the remaining terms. Obviously, a subsequence can also be infinite, if the original sequence is also infinite. More formally:

Definition 2.3 (Subsequence)**.** Given a sequence $\{v^t\}$, a subsequence of $\{v^t\}$ is another sequence, of the form $\{v^{t_j}\}$, such that $\{t_j\}$ is a strictly monotonically increasing sequence of natural numbers.

Before defining what a convergent sequence is, we need the notions of a metric space and a metric:

Definition 2.4 (Metric and Metric Space)**.** A *metric space* is a set V, together with a real–valued function $d : V \times V \to \mathbb{R}$ (called *metric*) such that, for all $v, v', v'' \in V$ the following properties hold:

- $d(v, v') \geq 0$ (non–negativity),
- $d(v, v') = 0$ if and only if $v = v'$ (identity of indiscernibles),

- $d(v, v') = d(v', v)$ (symmetry),
- $d(v, v'') \leq d(v, v') + d(v', v'')$ (triangle inequality).

Now we can establish what a convergent sequence is:

Definition 2.5 (Convergent Sequence)**.** A *convergent sequence* in a metric space V is a sequence $\{v^t\}$, whose terms lie in V and for which there exists a point $v \in V$ such that, for every real number $\epsilon > 0$, there exists a natural number T such that $d(v, v^t) < \epsilon$ for all $t > T$.

Intuitively, what this means is that the terms of the sequence get arbitrarily close to the point v, where the distance to this point is measured by the metric inherent to the metric space V. Such a point, if it exists, is called the *limit* (also *limit point*) of the sequence, and then the sequence is said to *converge* to v. In case it does not exist, then the sequence is called *divergent*. It is easy to prove that if a sequence converges, all its subsequences converge to the same limit.

Another special type of sequences are the so–called Cauchy sequences:

Definition 2.6 (Cauchy Sequence)**.** A *Cauchy sequence* in a metric space V is a sequence $\{v^t\}$, whose terms lie in V and such that, for every real number $\epsilon > 0$, there exists a natural number T such that, for all $t, t' > T$, we have $d(v_t, v_{t'}) < \epsilon$.

What this means is that the terms of a Cauchy sequence get arbitrarily close to each other, suggesting that the sequence is convergent. This is indeed the case, but the limit v need not lie in the metric space V.

Henceforth, we change the notation to have superscripts $\overrightarrow{\cdot}$, indicating that we are working with vector spaces V. A basic operation on vector spaces are norms:

Definition 2.7 (Norm)**.** Given a vector space V, a *norm* on V is a function $\|\cdot\| : V \to \mathbb{R}$ such that, for all $\overrightarrow{v}, \overrightarrow{v}' \in V$ and for all $\lambda \in \mathbb{R}$, the following properties hold:

- $\|\overrightarrow{v}\| \geq 0$ (positivity),
- $\|\overrightarrow{v}\| = 0$ if and only if $\overrightarrow{v} = \overrightarrow{0}$ (point separation),
- $\|\lambda\overrightarrow{v}\| = |\lambda| \, \|\overrightarrow{v}\|$ (positive homogeneity),
- $\|\overrightarrow{v} + \overrightarrow{v}'\| \leq \|\overrightarrow{v}\| + \|\overrightarrow{v}'\|$ (triangle inequality).

Notice the similarity of these properties and the ones for a metric in Definition 2.4. In fact, every norm $\|\cdot\|$ gives rise to a metric, by means of $d(\overrightarrow{v}, \overrightarrow{v}') = \|\overrightarrow{v} - \overrightarrow{v}'\|$. Using Definition 2.7 we get:

Definition 2.8 (Normed Space)**.** A *normed space* V is a vector space endowed with a norm.

Therefore, any normed space with the norm $\|\cdot\|$ is a metric space under the metric $d(\overrightarrow{v}, \overrightarrow{v}') = \|\overrightarrow{v} - \overrightarrow{v}'\|$.

Next, we move on to the concepts of bilinear form and inner product:

Definition 2.9 (Bilinear Form)**.** Given a vector space V, a *bilinear form* is a function $f : V \times V \to \mathbb{R}$ such that, for all $\overrightarrow{v}, \overrightarrow{v}', \overrightarrow{v}'' \in V$ and for all $\lambda, \lambda' \in \mathbb{R}$ the following is true:

- $f\left((\lambda\overrightarrow{v} + \lambda'\overrightarrow{v}'), \overrightarrow{v}''\right) = \lambda f(\overrightarrow{v}, \overrightarrow{v}'') + \lambda' f(\overrightarrow{v}', \overrightarrow{v}'')$.
- $f(\overrightarrow{v}'', (\lambda\overrightarrow{v} + \lambda'\overrightarrow{v}')) = \lambda f(\overrightarrow{v}'', \overrightarrow{v}) + \lambda' f(\overrightarrow{v}'', \overrightarrow{v}')$.

If it also satisfies $f(\overrightarrow{v}, \overrightarrow{v}') = f(\overrightarrow{v}', \overrightarrow{v})$, it is called *symmetric*.

Definition 2.10 (Inner Product)**.** Given a vector space V, an *inner product* (also *dot product*) is a symmetric bilinear form $\langle\cdot,\cdot\rangle$ that is positive definite, that is, $\langle\overrightarrow{v}, \overrightarrow{v}\rangle \geq 0$, with equality only for $\overrightarrow{v} = \overrightarrow{0}$.

Every inner product induces a corresponding norm via $\|\cdot\| = \sqrt{\langle\cdot,\cdot\rangle}$. Using now Definition 2.10 we have:

Definition 2.11 (Inner Product Space)**.** An *inner product space* (also *pre–Hilbert space* or *dot product space*) V is a vector space endowed with an inner product.

Hence, any inner product space is a normed space (and consequently a metric space) under the norm $\|\cdot\| = \sqrt{\langle\cdot,\cdot\rangle}$. Rereading Definition 1.16, the phrase "a feature space $\mathcal{H}$ endowed with a dot product", as we know now, can be rewritten as "an inner product (feature) space $\mathcal{H}$".

Next we discuss when Cauchy sequences converge in the same space, with the following definition:

Definition 2.12 (Complete Space)**.** A metric space V is said to be *complete* if every Cauchy sequence in V converges to a point that also belongs to V.

Combining Definitions 2.12 and 2.11, we get:

Definition 2.13 (Hilbert Space)**.** A Hilbert space $\mathcal{H}$ is a complete dot product space.

Notice that we have switched from notation V to $\mathcal{H}$. We do this in order to establish a connection with the feature space of Subsection 1.3.2. As will be seen in Section 2.2 in detail, this feature space $\mathcal{H}$ is indeed a Hilbert space, hence the notation $\mathcal{H}$.

Turning back to sequences, in this thesis we will deal with sequences in complete spaces, be it real (that is, with $V = \mathbb{R}^d$ in Definition 2.1), or in a Hilbert feature space $\mathcal{H}$ induced by a kernel function (that is, with $V = \mathcal{H}$).

Some basic and classical results are the following:

Proposition 2.14 (Convergent sequences are Cauchy)**.** *Every convergent sequence is a Cauchy sequence.*

Proposition 2.15 (Cauchy sequences are bounded)**.** *Every Cauchy sequence in a normed space endowed with the norm $\|\cdot\|$ is bounded, that is, there exists a value $R < \infty$ such that $\|v^t\| \leq R$ for all t.*

Proposition 2.16 (Convergent subsequences imply convergent Cauchy sequences)**.** *In a metric space, a Cauchy sequence that has a convergent subsequence is itself convergent, and it converges to the same limit than this subsequence.*

Using these propositions, we can formulate two important classical theorems:

Theorem 2.17 (Monotone Convergence Theorem)**.** *If a sequence of real numbers is monotone, the sequence has a finite limit if and only if the sequence is bounded.*

Proposition 2.18. *Every sequence of real numbers has a monotone subsequence.*

Theorem 2.19 (Bolzano–Weierstrass Theorem)**.** *Each bounded sequence in $\mathbb{R}^d$ has a convergent subsequence.*

This last theorem is proved with Proposition 2.18 by taking subsequences in different coordinates of the elements $\overrightarrow{v}^t \in \mathbb{R}^d$.

2.1.2 Rates of Convergence

So far we have only considered convergent sequences, but a key question, in order to compare iterative methods, is to check the speed at which a convergent sequence approaches its limit. Given two different methods that generate sequences approximating a desired limit, the best method is the one whose sequence approaches faster this limit.

In order to characterize this speed, the following definitions are used:

Definition 2.20 (Rates of Convergence)**.** Given a normed vector space V endowed with the norm $\|\cdot\|$, a sequence $\left\{\overrightarrow{v}^t\right\}$ in V is said to *converge linearly* to the limit $\overrightarrow{v}$ if there is a constant $0 < c < 1$ and a $T \in \mathbb{N}$ such that

$$\lim_{t\to\infty} \frac{\left\|\overrightarrow{v}^{t+1} - \overrightarrow{v}\right\|}{\left\|\overrightarrow{v}^{t} - \overrightarrow{v}\right\|} = c$$

for all $t > T$. If the above holds for $c = 0$ it *converges superlinearly*, and if it holds for $c = 1$ it *converges sublinearly.*

To distinguish different cases of superlinear convergence, we apply the following:

Definition 2.21 (Cases of Superlinear Convergence)**.** Given a normed vector space V endowed with the norm $\|\cdot\|$, a sequence $\left\{\overrightarrow{v}^t\right\}$ in V is said to *converge superlinearly with order* q to the limit $\overrightarrow{v}$, with $q > 1$, if there is a constant $0 < c < 1$ and a $T \in \mathbb{N}$ such that

$$\lim_{t\to\infty} \frac{\left\|\overrightarrow{v}^{t+1} - \overrightarrow{v}\right\|}{\left\|\overrightarrow{v}^{t} - \overrightarrow{v}\right\|^q} = c$$

for all $t > T$. If the above holds for $q = 2$ it *converges quadratically*, and if it holds for $q = 3$ it *converges cubically.*

We resume next the topic of kernels.

2.2 Kernel Functions

Subsection 1.3.2 introduced the notion of kernel function. There it was seen how its use makes possible to learn non–linear machines by using linear algorithms, as long as the only operations between two patterns are dot products. This is so because the kernel implicitly computes the dot product of the mapping of both patterns in the feature space, as expressed in (1.10). By substituting each dot product for the corresponding call to the kernel function, we circumvent the explicit expression of the projections of the patterns onto the feature space.

Note, however, that this shortcut would be useless if, in order to find a kernel function, we needed first to create a complicated feature space $\mathcal{H}$, then define the dot product operation in this space, and finally set the kernel as the computation of this dot product. In fact, there is no need to know explicitly which is the expression of the feature map $\Phi(\cdot)$, provided that the kernel function satisfies some properties that are explained next.

2.2.1 Properties of Kernel Functions

We begin with some obvious properties. The first one, according to (1.10), is that the kernel function must be symmetric, that is, $k(\overrightarrow{x}, \overrightarrow{x}') = k(\overrightarrow{x}', \overrightarrow{x})$, since the dot product is indeed symmetric.

In order to introduce the next property, we need some definitions first. Recall that it will always be assumed that the input space $\mathcal{X}$ satisfies $\mathcal{X} \subseteq \mathbb{R}^d$.

Definition 2.22 (Kernel Matrix)**.** Given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a training set $\{\overrightarrow{x}_i : i = 1, \ldots, N\}$, the *kernel matrix* (also Gram matrix) is the $N \times N$ matrix K with elements $K_{ij} = k(\overrightarrow{x}_i, \overrightarrow{x}_j)$, with $i, j = 1, \ldots, N$.

Definition 2.23 (Positive Definite Matrix)**.** A real symmetric $N \times N$ matrix A is *positive definite* (respectively, *negative definite*) if, for every vector $\overrightarrow{x} \neq \overrightarrow{0}$ in $\mathbb{R}^d$, we have $\langle \overrightarrow{x}, A\overrightarrow{x} \rangle > 0$ (< 0). It is called *positive semidefinite* (respectively, *negative semidefinite*) if, for every vector $\overrightarrow{x} \in \mathbb{R}^d$, we have $\langle \overrightarrow{x}, A\overrightarrow{x} \rangle \geq 0$ (≤ 0).

Building on the previous two definitions, the next one is:

Definition 2.24 (Positive (Semi)Definite Kernel)**.** Given an input space $\mathcal{X}$, a *positive (semi)definite kernel* is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that, for all N and for every training set $\{\overrightarrow{x}_i : i = 1, \ldots, N\}$, the associated kernel matrix with elements $K_{ij} = k(\overrightarrow{x_i}, \overrightarrow{x_j})$, with $i, j = 1, \ldots, N$, is positive (semi)definite.

We can state now the following result, recovering Definition 1.16:

Proposition 2.25 (Requirements for a Kernel)**.** *Given a finite input space $\mathcal{X}$, a dot product feature space $\mathcal{H}$, and a (feature) map $\Phi : \mathcal{X} \to \mathcal{H}$, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel (that is, $k(\overrightarrow{x}, \overrightarrow{x}') = \langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x}') \rangle$ for every $\overrightarrow{x}, \overrightarrow{x}' \in \mathcal{X}$) if and only if it is a positive semidefinite kernel.*

This is the reason why in the literature the term *kernel* is the short–hand for *positive semidefinite kernel*, and they are interchangeable. As for the term *kernel* itself, the word comes from integral operator theory, which was studied, among others, by Mercer [32] and Hilbert [33].

By definition, the property of being a positive semidefinite kernel ensures symmetry. Another consequence is that $k(\overrightarrow{x}, \overrightarrow{x}) \geq 0$, which is coherent with the facts that $k(\overrightarrow{x}, \overrightarrow{x}) = \|\phi(\overrightarrow{x})\|_2^2$ and that a norm is non–negative.

It remains to see why any positive semidefinite kernel is equivalent to a dot product in the feature space, because so far we have taken this for granted. In the next section, we explicitly construct a (Hilbert) space such that its dot product is indeed the kernel.

2.2.2 Reproducing Kernel Hilbert Spaces

Let us consider a given (positive semidefinite) kernel k, and a non–empty set $\mathcal{X} \subset \mathbb{R}^d$. If we define explicitly the feature map as

$$\Phi\left(\overrightarrow{x}\right) = k\left(\overrightarrow{x}, \cdot\right),$$

we have that Φ is a function that maps from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$. In other words, given a pattern $\overrightarrow{x}$, the map $\Phi\left(\overrightarrow{x}\right)$ transforms it into a function with domain $\mathcal{X}$. This function expresses its similarity to other patterns, because it is given by the kernel k.

At this point, according to Definition 2.11, we need to construct a dot product space $\mathcal{H}$ such that the kernel k performs this dot product. One possibility for $\mathcal{H}$ is what is called the *Reproducing Kernel Hilbert Space* (RKHS). To find the RKHS, we will proceed in two steps: first, constructing the vector space itself, and secondly, endowing it with a dot product.

For the first part, we need to recall the concept of (linear) span:

Definition 2.26 (Linear Span)**.** Given a vector space V, and $S = \{\overrightarrow{v}_1, \ldots, \overrightarrow{v}_N\}$ a fixed set of $N \in \mathbb{N}$ vectors belonging to V, the *linear span* (or simply *span*) of S is the set formed by all linear combinations taken from S.

Thus, the space $\mathcal{H}$ is constructed by taking linear combinations of the form

$$f\left(\cdot\right) = \sum_{i=1}^{N} \lambda_i \Phi\left(\overrightarrow{x}_i\right) = \sum_{i=1}^{N} \lambda_i k\left(\overrightarrow{x}_i, \cdot\right),$$

where $N \in \mathbb{N}$ and $\overrightarrow{x}_1, \ldots, \overrightarrow{x}_N \in \mathcal{X}$ are arbitrary. Given f and another function $f' = \sum_{j=1}^{N'} \lambda'_j k\left(\overrightarrow{x}'_j, \cdot\right)$, the dot product is defined as

$$\left\langle f, f' \right\rangle = \sum_{i=1}^{N} \sum_{j=1}^{N'} \lambda_i \lambda'_j k\left(\overrightarrow{x}_i, \overrightarrow{x}'_j\right).$$

Making use of the symmetry of the kernel, we have that

$$\left\langle f, f' \right\rangle = \sum_{j=1}^{N'} \lambda'_j f\left(\overrightarrow{x}'_j\right) = \sum_{i=1}^{N} \lambda_i f'\left(\overrightarrow{x}_i\right).$$

As a result, $\langle \cdot, \cdot \rangle$ is bilinear. It is also symmetric, since $\langle f, f' \rangle = \langle f', f \rangle$. Besides, it is non–negative, as

$$\langle f, f \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j k\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) \geq 0,$$

where the last inequality follows from Proposition 2.25 and the fact that we assume the input space $\mathcal{X}$ to be a subset of $\mathbb{R}^d$.

According to Definition 2.10, it only remains to see that $\langle f, f \rangle = 0$ implies $f = 0$. To see this, we need to notice that $\langle k\left(\overrightarrow{x}, \cdot\right), f \rangle = f\left(\overrightarrow{x}\right)$. In particular, $\langle k\left(\overrightarrow{x}, \cdot\right), k\left(\overrightarrow{x}', \cdot\right) \rangle = k\left(\overrightarrow{x}, \overrightarrow{x}'\right)$. This last property is called the *reproducing property* of kernels, so sometimes positive semidefinite kernels are also called *reproducing kernels*.

Because we have endowed the space $\mathcal{H}$ with an inner product, we can apply the Cauchy–Schwarz inequality to get:

$$f\left(\overrightarrow{x}\right)^2 = \langle k\left(\overrightarrow{x}, \cdot\right), f \rangle^2 \leq k\left(\overrightarrow{x}, \overrightarrow{x}\right) \langle f, f \rangle,$$

and then $\langle f, f \rangle = 0$ implies $f = 0$. Finally, the reproducing property of the kernel k, together with the previous definition of the feature map Φ, mean that

$$\left\langle \Phi\left(\overrightarrow{x}\right), \Phi\left(\overrightarrow{x}'\right) \right\rangle = \left\langle k\left(\overrightarrow{x}, \cdot\right), k\left(\overrightarrow{x}', \cdot\right) \right\rangle = k\left(\overrightarrow{x}, \overrightarrow{x}'\right),$$

satisfying thus Definition 1.16.

Here we have seen that any positive semidefinite kernel k can be regarded as a dot product in the inner product space built above. However, we will see later that this is not the only possible feature space associated to this particular kernel.

For the moment, let us consider the (seldom seen in practice) opposite case, where we start from an already known feature map $\Phi(\cdot)$ to a dot product space. Building the positive semidefinite kernel is then trivial by $k\left(\overrightarrow{x}, \overrightarrow{x}'\right) = \langle \Phi\left(\overrightarrow{x}\right), \Phi\left(\overrightarrow{x}'\right) \rangle$. The kernel is positive semidefinite because for all $\lambda_i \in \mathbb{R}$, $\overrightarrow{x}_i \in \mathcal{X}$, $i = 1, \ldots, N$, we get, due to the non–negativity of the norm, that

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j k\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) = \left\langle \sum_{i=1}^{N} \lambda_i \Phi\left(\overrightarrow{x}_i\right), \sum_{j=1}^{N} \lambda_i \Phi\left(\overrightarrow{x}_j\right) \right\rangle = \left\| \sum_{i=1}^{N} \lambda_i \Phi\left(\overrightarrow{x}_i\right) \right\|^2 \geq 0.$$

In order to switch from the inner product space constructed above to a Hilbert space, according to Definition 2.13, we need to complete it. This is done mathematically, among other things (further details are beyond the scope of this work), by adding the limit points of Cauchy sequences to the space, where the convergence of such sequences is measured in this case with the norm $\|f\| = \sqrt{\langle f, f\rangle}$ [2]. Such a procedure is called *completion*.

With all this is mind, we arrive to the definition of *Reproducing Kernel Hilbert Space (RKHS)*:

Definition 2.27 (Reproducing Kernel Hilbert Space (RKHS))**.** Given a nonempty set $\mathcal{X}$, a *reproducing kernel Hilbert space* is a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, endowed with the dot product $\langle f, f\rangle$ and its corresponding norm $\|f\| = \sqrt{\langle f, f\rangle}$, such that there exists a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the properties:

- $\langle f, k(\overrightarrow{x}, \cdot)\rangle = f(\overrightarrow{x})$ for all $f \in \mathcal{H}$ (reproducing property),
- $\mathcal{H}$ is the completion of the linear span of the functions $k(\overrightarrow{x}, \cdot)$, with $\overrightarrow{x} \in \mathcal{X}$.

The RKHS for a given kernel is unique, as the following theorem states [34]:

Theorem 2.28 (Moore–Aronszajn Theorem)**.** *Given a nonempty set $\mathcal{X}$ and a symmetric positive semidefinite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there is a unique Hilbert space of functions $\mathcal{H}$ for which k is a reproducing kernel.*

2.2.3 Mercer Kernels

Even if the RKHS for a given kernel is unique, this RKHS is not the only possible feature space associated to a kernel. For instance, there is another possibility called the Mercer map, arising from Mercer theorem in [32]. To understand this theorem, we need some preliminary definitions:

Definition 2.29 ((Real) L^p–Space)**.** Given a subset $\mathcal{X}$ of $\mathbb{R}^d$, the *space* $L^p(\mathcal{X})$ is the set of continuous real–valued functions f for which the integral is finite, that its, $\|f\|_{L^p} = \left(\int_{\mathcal{X}} |f(\overrightarrow{x})|^p \, d\overrightarrow{x}\right)^{\frac{1}{p}} < \infty$.

If $p = 2$ we obtain a Hilbert space, with the inner product given by

$$\langle f, f'\rangle = \int_{\mathcal{X}} f(\overrightarrow{x}) \, f'(\overrightarrow{x}) \, d\overrightarrow{x}.$$

The concepts of l_p–norm and l_p–space are closely related to that of L_p–space. We begin with the norm:

Definition 2.30 (l_p–Norm)**.** Given a subset $\mathcal{X}$ of $\mathbb{R}^d$, the *l_p–norm* is a norm, defined for $\overrightarrow{x}$ in $\mathcal{X}$ and $p \geq 1$ as

$$\|\overrightarrow{x}\|_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{\frac{1}{p}}.$$

In the limit $p = \infty$, we have $\|\overrightarrow{x}\|_\infty = \max_{i=1,\ldots,d}\{|x_i|\}$.

The l_p–space is defined by extending the l_p–norm to measure sequences, in the following way:

Definition 2.31 ((Real) l_p–Space)**.** The *l_p–space* is the set of all sequences of real numbers $\left\{x^t\right\}_{t=1}^{n}$, with $N \in \mathbb{N}$, for which the l_p–norm is finite, that is:

$$\left\|\left\{x^t\right\}_{t=1}^{N}\right\|_p = \left(\sum_{t=1}^{N} |x^t|^p\right)^{\frac{1}{p}} < \infty.$$

The above definition also covers the case of infinite sequences with $N = \infty$. As for the L_p–space, the l_2 space is a Hilbert space, where the inner product of two sequences $\left\{x^t\right\}_{t=1}^{N}$ and $\left\{y^t\right\}_{t=1}^{N}$ is defined as $\langle x, y\rangle = \sum_{t=1}^{N} x^t y^t$.

Some other required definitions are:

Definition 2.32 ((Real) Linear Map)**.** Given two vector spaces V, V^*, a *linear map* (also *linear operator*, *linear transformation* or *linear function*) is a function $A : V \to V^*$, such that, for every $\overrightarrow{v}, \overrightarrow{v}' \in V$ and every $\lambda, \lambda' \in \mathbb{R}$, it holds that $A\left(\lambda\overrightarrow{v} + \lambda'\overrightarrow{v}'\right) = \lambda A\left(\overrightarrow{v}\right) + \lambda' A\left(\overrightarrow{v}'\right)$.

Definition 2.33 (Eigenvalue and Eigenfunction)**.** Given a linear map A over a vector space of functions V, an *eigenfunction* f is any non–zero function in V that is unchanged by A, except possibly for a multiplicative scaling factor, called its *eigenvalue*. More precisely, $Af = \lambda f$, where λ is the eigenvalue of the eigenfunction f.

Definition 2.34 (Compact Set)**.** A subset $\mathcal{X}$ of $\mathbb{R}^d$ is said to be *compact* if every sequence in X has a convergent subsequence whose limit is also in $\mathcal{X}$.

Now we are ready to state a simplified version of Mercer's theorem (further details are beyond the scope of this work):

Theorem 2.35 (Mercer Theorem)**.** *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^d$ and k a continuous symmetric kernel such that the integral linear operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$ given by*

$$\left(T_k f\right)\left(\overrightarrow{x}\right) = \int_{\mathcal{X}} k\left(\overrightarrow{x}, \cdot\right) f\left(\overrightarrow{x}\right) \, d\overrightarrow{x},$$

is positive semidefinite, which means in this context that

$$\int_{\mathcal{X}\times\mathcal{X}} k\left(\overrightarrow{x}, \overrightarrow{x}'\right) f\left(\overrightarrow{x}\right) f\left(\overrightarrow{x}'\right)\, d\overrightarrow{x} d\overrightarrow{x}' \geq 0,$$

for every $f \in L_2(\mathcal{X})$*. We can then expand the kernel via*

$$k\left(\overrightarrow{x}, \overrightarrow{x}'\right) = \sum_{i=1}^{\infty} \lambda_i \psi_i\left(\overrightarrow{x}\right) \psi_i\left(\overrightarrow{x}'\right),$$

where $\psi_i(\cdot) \in L_2(\mathcal{X})$ *are eigenfunctions of* T_k*, normalized in such a way that* $\|\psi_i\|_{L_2} = 1$*, with* λ_i *their associated eigenvalues.*

If a kernel satisfies the above, it is called a *Mercer kernel*. This term is an alternative to positive semidefinite kernel or reproducing kernel. In fact, if we construct the feature map $\Phi : \mathcal{X} \to l_2$ given by

$$\Phi(\cdot) = \left\{\sqrt{\lambda_j}\psi_j(\cdot)\right\}_{j=1}^{\infty},$$

we will have that

$$\left\langle \Phi\left(\overrightarrow{x}\right), \Phi\left(\overrightarrow{x}'\right)\right\rangle = \sum_{j=1}^{\infty} \lambda_j \psi_j\left(\overrightarrow{x}\right) \psi_j\left(\overrightarrow{x}'\right) = k\left(\overrightarrow{x}, \overrightarrow{x}'\right),$$

and thus it is readily seen that $\Phi(\cdot)$ is also a valid feature map for the kernel k.

In summary, we have seen that a positive semidefinite kernel k can be regarded as a dot product in two different associated feature spaces: the RKHS and the Mercer space. Moreover, these are not necessarily the only possible feature spaces for k (in fact, the different choices may even differ in their dimensionality $d_{\mathcal{H}}$) [2].

In any case, this is rarely important in practice, because, by the kernel trick, we actually do not care about what the feature space is. As we will see, all the operations between vectors in SVMs are dot products, so using a kernel ensures that we are implicitly working in one of those feature spaces, but it is not relevant to know in which particular space.

2.2.4 Examples of Kernels

Now that we have a theoretic knowledge of kernels, we show some concrete examples of functions that can be used as kernels. We begin with the polynomial kernel, which has two different variants:

Definition 2.36 (Polynomial Kernel)**.** Given $\mathcal{X} \subseteq \mathbb{R}^d$, and $\vec{x}, \vec{x}' \in \mathcal{X}$, the *homogeneous polynomial kernel* is calculated as

$$k\left(\vec{x}, \vec{x}'\right) = \left\langle \vec{x}, \vec{x}' \right\rangle^p,$$

with $p \in \mathbb{N}$. The *inhomogeneous polynomial kernel* is given by

$$k\left(\vec{x}, \vec{x}'\right) = \left(\left\langle \vec{x}, \vec{x}' \right\rangle + c\right)^p,$$

with $d \in \mathbb{N}$ and $c \geq 0$.

The homogeneous polynomial kernel with $p = 1$ is usually called the *linear kernel*, since it reduces to the standard dot product.

However, the most popular kernel function nowadays is the Gaussian radial basis function kernel, abbreviated as the Gaussian kernel:

Definition 2.37 (Gaussian kernel)**.** Given $\mathcal{X} \subseteq \mathbb{R}^d$, and $\vec{x}, \vec{x}' \in \mathcal{X}$, the *Gaussian Kernel* (also *RBF kernel*) is calculated as

$$\exp\left(-\frac{\|\vec{x} - \vec{x}'\|_2^2}{\sigma^2}\right),$$

where $\sigma > 0$.

Recalling the concepts of linear independence and rank of a matrix, this kernel is very convenient because of a result stated below [35]:

Theorem 2.38 (Full Rank of Gaussian Kernel)**.** *Given a training set* $\{\vec{x}_i, i = 1, \ldots, N\}$ *and* $\sigma \neq 0$*, the kernel matrix obtained with the Gaussian kernel has full rank, provided that* $\vec{x}_1, \ldots, \vec{x}_N$ *are distinct.*

Intuitively, this means that the vectors $\Phi\left(\vec{x}_1\right), \ldots, \Phi\left(\vec{x}_N\right)$ are linearly independent, so they occupy a space as large as possible. In terms of the operator of Theorem 2.35, it has countably many eigenvectors and eigenfunctions, so $d_{\mathcal{H}} = \aleph_0 = |\mathbb{N}|$.

Recalling Figure 1.11, this in theory increases the options of finding a separating hyperplane.

We can also build some more complex kernels from basic ones, such as the ones included in the following proposition:

Proposition 2.39 (Building kernels)**.** *Given two kernels k and k' over $\mathcal{X} \subseteq \mathbb{R}^d$, $\overrightarrow{x}, \overrightarrow{x}'$ whichever two vectors in $\mathcal{X}$, $c > 0$ and $f : \mathcal{X} \to \mathbb{R}$, the following are also kernels:*

- $k\left(\overrightarrow{x}, \overrightarrow{x}'\right) + k'\left(\overrightarrow{x}, \overrightarrow{x}'\right)$,
- $ck\left(\overrightarrow{x}, \overrightarrow{x}'\right)$,
- $k\left(\overrightarrow{x}, \overrightarrow{x}'\right) k'\left(\overrightarrow{x}, \overrightarrow{x}'\right)$,
- $f\left(\overrightarrow{x}\right) f\left(\overrightarrow{x}'\right)$,
- $\exp\left(k\left(\overrightarrow{x}, \overrightarrow{x}'\right)\right)$.

However, in this thesis we will stick to the basic kernels enumerated above.

2.3 Convex Geometry

We switch now to a different topic, that of convex geometry. Before moving on to the topic of optimization, we need to understand the concept and properties of convex sets. After that, we introduce convex hulls, which will be fundamental in Chapter 3 as a reinterpretation of SVMs.

The concept of convex set is easy to define:

Definition 2.40 (Convex Set)**.** A set $\mathcal{X}$ in a vector space V is called *convex* if the closed line segment between any two points $\overrightarrow{x}, \overrightarrow{x}'$ in $\mathcal{X}$ lies itself in $\mathcal{X}$.

Visually, a set is convex if every point in the set can be "seen" by every other point in the set, along a straight path between them which does not leave the set. This is illustrated in Figure 2.1, where a convex and a non–convex set are depicted.

A special kind of points in a convex set are its extreme points:

Definition 2.41 (Extreme Point)**.** Given a convex set $\mathcal{X}$ in $\mathbb{R}^d$, an *extreme point* of $\mathcal{X}$ is a point which does not lie in any open line segment between two points of $\mathcal{X}$.

Visually, the extreme points are the "boundary points" of the convex set, as illustrated in Figure 2.2. In the case of a curve border, all the points in the curve are extreme, whereas in the case of straight borders, the extreme points are the "vertices" or the "corners" of the set.

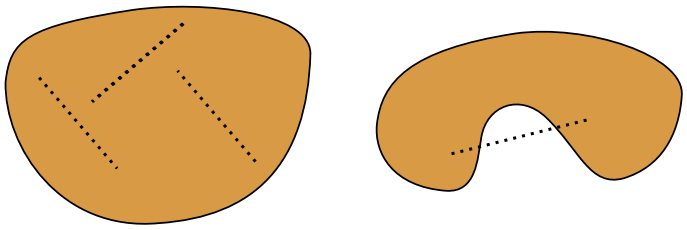

FIGURE 2.1: Illustration of a convex (left) and a non-convex set (right). The dashed lines show that in the convex one, no matter which line segment we pick, it will always lie in the set, whereas in the non-convex one this is not true for the segment depicted.

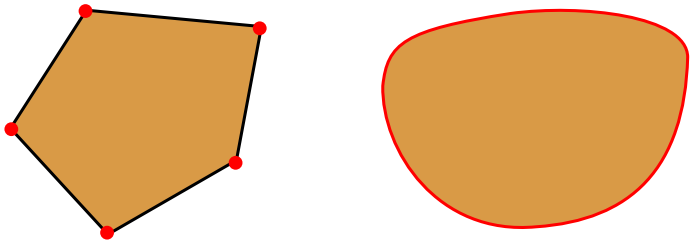

FIGURE 2.2: Illustration of the extreme points (in red) of two convex sets. In the first set the extreme points are the vertices of the polygon, whereas in the second set they cover the border of the set.

Recalling that a convex combination is a linear combination where the weights sum up to 1, now we can define convex hulls:

Definition 2.42 (Convex Hull)**.** Given a set $\mathcal{X}$ in a vector space V, the *convex hull* of $\mathcal{X}$ is the set of all convex combinations of points in $\mathcal{X}$, that is, the set:

$$\left\{ \overrightarrow{x} : \overrightarrow{x} = \sum_{i=1}^{N} \lambda_i \overrightarrow{x}_i,\ N \in \mathbb{N},\ \overrightarrow{x}_i \in \mathcal{X}\ \forall i,\ \lambda_i \geq 0\ \forall i, \sum_{i=1}^{N} \lambda_i = 1 \right\}.$$

We will use the notation $\text{conv}\,(\mathcal{X})$ to refer to the convex hull of $\mathcal{X}$. It is easily seen that $\text{conv}\,(\mathcal{X})$ is a convex set. In fact, it is the smallest convex set that contains $\mathcal{X}$, as illustrated in Figure 2.3. A comparison that is usually used is the one of an elastic band that contracts from infinity and tightens around the set.

An important historical result is the following [36]:

Theorem 2.43 (Krein–Milman Theorem)**.** *A compact convex set $\mathcal{X}$ in a normed vector space $\mathcal{V}$ is the convex hull of its extreme points.*

Thus, in the particular case of a training set of the form (1.1), the convex hull of the instances $\overrightarrow{x}_1, \ldots, \overrightarrow{x}_N$ is completely determined by its extreme points.

Considering from now on this finite case, another special point is the barycenter:

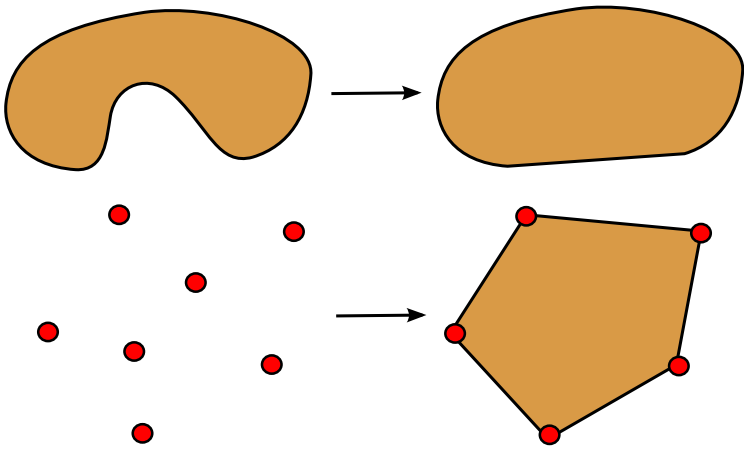

FIGURE 2.3: Convex hulls of two different sets. In the first case, the kidney–shaped set of Figure 2.1 is made convex with minimal surface by its convex hull. In the second case, a set of points is enclosed by the convex hull, yielding the polygon of Figure 2.2.

Definition 2.44 (Barycenter)**.** Given a set $\mathcal{X} = \{\overrightarrow{x}_1, \ldots, \overrightarrow{x}_N\}$, the barycenter of the convex hull of $\mathcal{X}$ is the point $\frac{1}{N}\sum_{i=1}^{N} \overrightarrow{x}_i$.

Therefore, the barycenter is the point obtained by taking the convex combination of the points of $\mathcal{X}$ with all coefficients equal to $\frac{1}{N}$.

Note that, in view of Definition 2.42, the coefficients λ_i satisfy $0 \leq \lambda_i \leq 1$. If we further restrict them, so that $0 \leq \lambda_i \leq \mu$, we obtain the following [24, 37]:

Definition 2.45 (Reduced Convex Hull)**.** Given a set $\mathcal{X}$ in a vector space V, the *reduced convex hull* of $\mathcal{X}$ with coefficient $\mu > 0$, is the set of all convex combinations of points in $\mathcal{X}$ with coefficients upper bounded by μ, that is, the set

$$\left\{ \overrightarrow{x} : \overrightarrow{x} = \sum_{i=1}^{N} \lambda_i \overrightarrow{x}_i,\ N \in \mathbb{N},\ \overrightarrow{x}_i \in \mathcal{X}\ \forall i,\ 0 \leq \lambda_i \leq \mu\ \forall i, \sum_{i=1}^{N} \lambda_i = 1 \right\}.$$

Analogously to the standard convex hull, we will use the notation $\text{rconv}_\mu(\mathcal{X})$. In our finite case where $\mathcal{X} = \{\overrightarrow{x}_1, \ldots, \overrightarrow{x}_N\}$, we have four different situations:

- $\mu < \frac{1}{N}$: then $\text{rconv}_\mu(\mathcal{X}) = \emptyset$, because the condition $\sum_{i=1}^{N} \lambda_i = 1$ cannot be met.
- $\mu = \frac{1}{N}$: in this case $\text{rconv}_\mu(\mathcal{X})$ reduces to the barycenter.
- $\frac{1}{N} < \mu < 1$: the reduced convex hull gradually expands from the barycenter to the convex hull, as μ increases (see Figure 2.4).
- $\mu \geq 1$: then $\text{rconv}_\mu(\mathcal{X}) = conv(\mathcal{X})$, because the coefficients must still sum up to 1 and be non–negative.

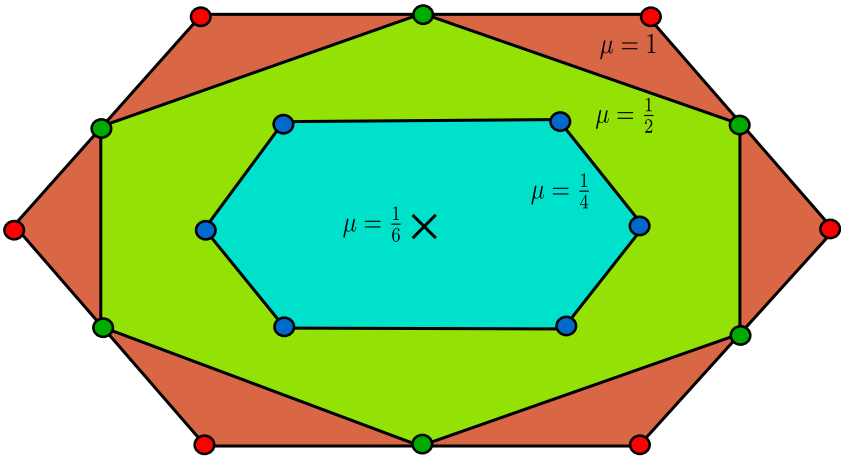

FIGURE 2.4: Evolution of a reduced convex hull as μ increases. The original convex hull is recovered when $\mu = 1$. Since there are 6 points, when $\mu = \frac{1}{6}$ it reduces to the barycenter, marked with a cross. Between these two values it gradually expands from the barycenter to the standard convex hull.

2.4 Optimization Theory

We are ready now for delving into optimization theory. Here we will see what kind of optimization problems we will have to cope with in the subsequent chapters. We will also introduce the concept of duality, which is vital in SVMs. Finally, we will see which are the optimality conditions of such problems, which will be useful for stopping the optimization process.

2.4.1 Convex Functions

To begin with, let us define what a convex function is.

Definition 2.46 (Convex Function)**.** Given a convex set $\mathcal{X}$, a function $f : \mathcal{X} \to \mathbb{R}$ is called itself *convex* if, for every $\overrightarrow{x}, \overrightarrow{x}'$ in $\mathcal{X}$ and their associated open line segment $\lambda\overrightarrow{x} + (1-\lambda)\overrightarrow{x}'$, with $0 < \lambda < 1$, it holds that

$$f\left(\lambda\overrightarrow{x} + (1-\lambda)\overrightarrow{x}'\right) \leq \lambda f\left(\overrightarrow{x}\right) + (1-\lambda) f\left(\overrightarrow{x}'\right).$$

If the above inequality is strict whenever $\overrightarrow{x} \neq \overrightarrow{x}'$, the function is said to be *strictly convex*.

Note that, because $\mathcal{X}$ is convex, the line segment still lies in $\mathcal{X}$. An example of such a function is seen in Figure 2.5, together with a non–convex function.

The complementary of a convex function is a concave function:

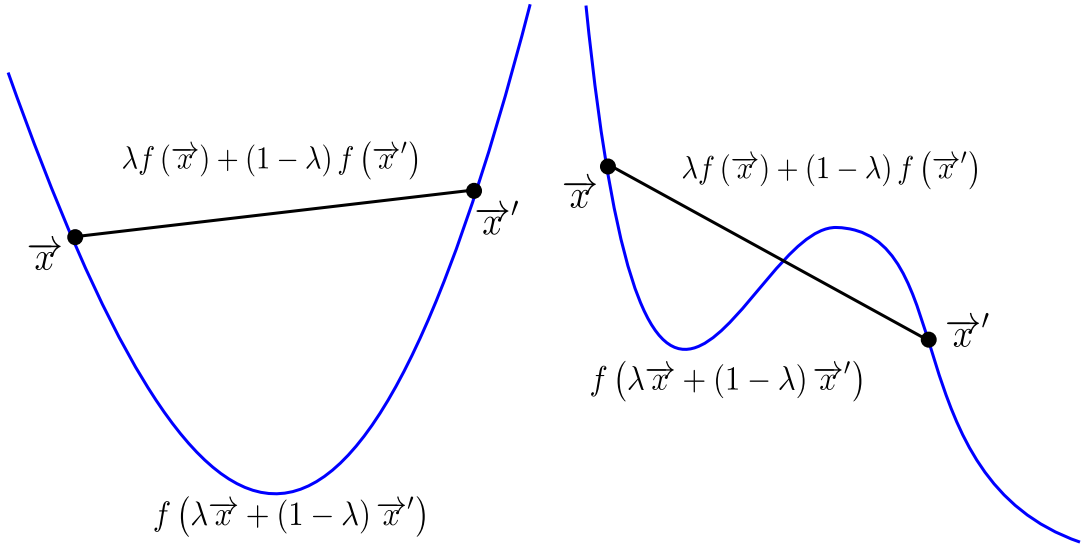

FIGURE 2.5: Example of a convex (left) and a non–convex function (right). Whereas in the convex case the value of the function is always below the line segment, this is not always true for a non–convex function.

Definition 2.47 ((Strictly) Concave Function)**.** Given a convex set $\mathcal{X}$, a function $f : \mathcal{X} \to \mathbb{R}$ is called *(strictly) concave* if $-f$ is a (strictly) convex function.

In order to optimize any function, we need to make clear the concept of minima and maxima of a function. There are two kinds of minima and maxima: either global or local. Beginning with the global ones:

Definition 2.48 (Global minimum (maximum))**.** A function $f : \mathcal{X} \to \mathbb{R}$ is said to have a *global minimum (maximum)* at $\overrightarrow{x}^* \in \mathcal{X}$ if $f(\overrightarrow{x}^*) \leq (\geq) f(\overrightarrow{x})$ for all $\overrightarrow{x} \in \mathcal{X}$.

Note that, since the inequality is not strict in this definition, the global minimum (maximum), if it exists, can be unique or not, as illustrated in Figure 2.6.

As for the local ones:

Definition 2.49 (Local minimum (maximum))**.** Given a normed space V endowed with the norm $\|\cdot\|$ and a subset $\mathcal{X} \subseteq V$, a function $f : \mathcal{X} \to \mathbb{R}$ is said to have a *local minimum (maximum)* at $\overrightarrow{x}^* \in \mathcal{X}$ if there exists an $\epsilon > 0$ such that $f(\overrightarrow{x}^*) \leq (\geq) f(\overrightarrow{x})$ for all $\overrightarrow{x} \in \mathcal{X}$ such that $\|\overrightarrow{x} - \overrightarrow{x}^*\| < \epsilon$.

Observe here that we need a normed space so that we can measure "distances". The idea is that a local minimum (maximum) is a minimum (maximum) of the function in a neighborhood given by its norm. If this neighborhood covers the whole domain of the function, then we get a global minimum (maximum). An illustration of this can be seen in Figure 2.7.

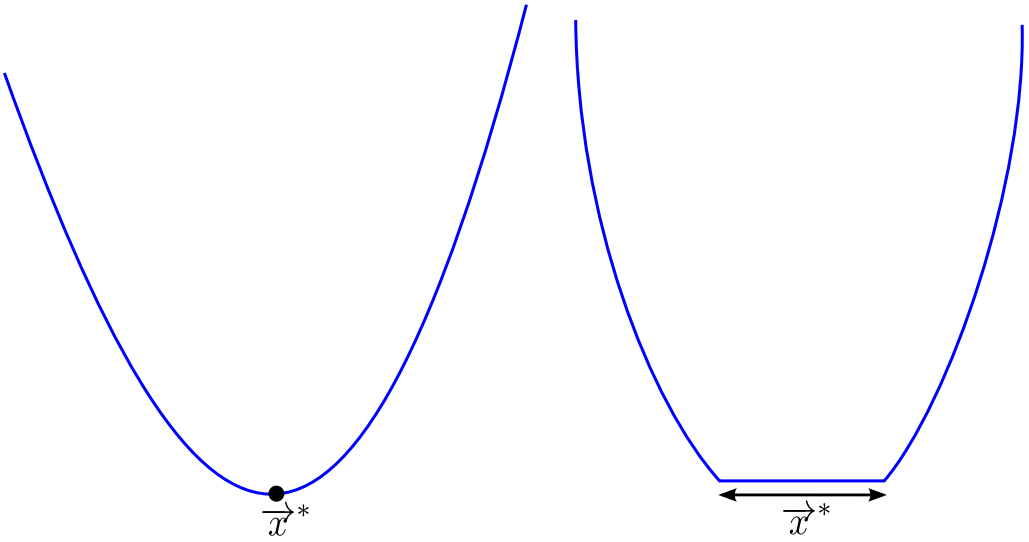

FIGURE 2.6: Example of a convex function with a unique global minimum (left) and of another convex function with a non–unique global minimum (right).

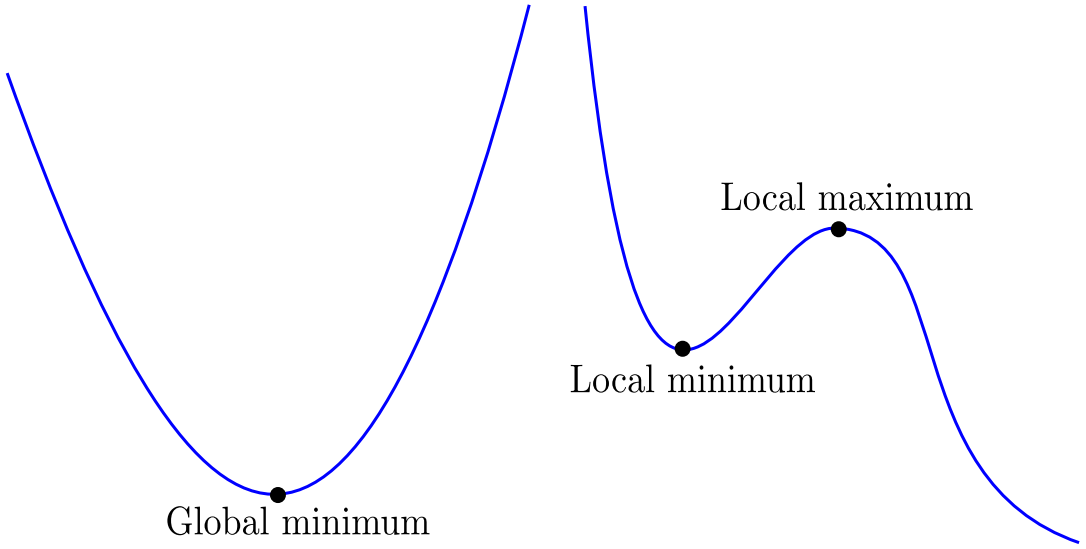

FIGURE 2.7: Illustration of global and local optimal points. Whereas "global" implies optimality in the whole domain, "local" only implies optimality in a neighborhood around the local optimum.

As we mentioned in Subsection 1.2.4, convex functions are very convenient for minimizing. This is also intuitive having at look at Figure 2.5, where we see that a convex function is "U–shaped". Illustrating this, we have the following result [38]:

Theorem 2.50 (Minima of a Convex Function)**.** *If a convex function $f : \mathcal{X} \to \mathbb{R}$ has a global minimum, then the subset of $\mathcal{X}$ of points where this minimum is attained is itself convex. Moreover, if f is strictly convex, this subset will only have a single point $\overrightarrow{x}^*$.*

Looking back at the examples in Figure 2.6, the function on the left is strictly convex and the one on the right is convex, but not strictly convex.

The subset of minima will be denoted from now on $\arg\min_{\overrightarrow{x}} f(\overrightarrow{x})$. In the case of a maximum, we will use the symbol $\arg\max$. In this theorem we can explicitly say "global minimum" because every local minimum of a convex function is itself a global minimum, due to the inherent convexity of $\mathcal{X}$. This is also seen in Figure 2.7: if there are local optima, then the function must be non–convex.

There is also a very important result regarding maximization of a convex function, due to Rockafellar [39], which makes use of Definition 2.41:

Theorem 2.51 (Maxima of a Convex Function)**.** *Given a compact convex set $\mathcal{X}$ and a convex function $f : \mathcal{X} \to \mathbb{R}$, the maximum of f over $\mathcal{X}$ is attained on the set of extreme points of $\mathcal{X}$.*

We can now state what do we mean by an optimization problem in our context, as well as some related terminology:

Definition 2.52 (Optimization Problem)**.** Given a function $f : \mathcal{X} \to \mathbb{R}$, a *minimization (maximization) problem* is the problem consisting of finding a global minimum (maximum) $\overrightarrow{x}^*$ of f in $\mathcal{X}$. The function f is termed *objective function* (also *cost function*), whereas the set $\mathcal{X}$ is called *feasible region* or *search space*. Any specific point $\overrightarrow{x} \in \mathcal{X}$ evaluated in the search is called a *feasible point* or a *candidate solution*.

2.4.2 Convex and Quadratic Programming

Depending on how we determine the feasible region we obtain different instances of optimization problems. Recall that in our context we will work in Euclidean space $\mathbb{R}^d$, where d will vary depending on the case. In the particular case where $\mathcal{X} = \mathbb{R}^d$, we obtain a so–called *unconstrained optimization problem*. This kind of problems are quite easy to deal with and have been studied in great depth: see [38, 40] for comprehensive references.

However, here we are interested in constrained optimization problems, which we define next:

Definition 2.53 (Constrained Optimization Problem)**.** Given real–valued functions f, $g_i, i = 1, \ldots, m$, $h_i, i = 1, \ldots, p$, with domain $\mathbb{R}^d$, a *constrained minimization problem in*

standard form is the minimization problem formulated as follows:

$$\begin{aligned} \min_{\overrightarrow{x}} \quad & f\left(\overrightarrow{x}\right) \\ \text{s.t.} \quad & \begin{cases} g_i\left(\overrightarrow{x}\right) \leq 0, & i = 1, \ldots, m, \\ h_i\left(\overrightarrow{x}\right) = 0, & i = 1, \ldots, p. \end{cases} \end{aligned}$$

The functions $g_i, i = 1, \ldots, m$ are called *inequality constraint functions* and the functions $h_i, i = 1, \ldots, p$ are the *equality constraint functions*. The standard form enforces that the right–hand sides of the constraints are always 0 and that the inequality constraints are always of type $\leq$. In the problems of subsequent chapters, we will encounter problems that are usually formulated with constraints of type $\geq$, but these can always be transformed into problems in standard form.

Together, the inequality and equality constraints define the feasible region $\mathcal{X} \subseteq \mathbb{R}^d$. The standard constrained maximization problem is identical, except for a max replacing the min.

Before explaining some particular instances of constrained optimization problems, we need the following definition:

Definition 2.54 ((Real) Affine Function)**.** An *affine function* is a function $f : \mathbb{R}^d \to \mathbb{R}^m$ that is expressible in the form

$$f\left(\overrightarrow{x}\right) = S\overrightarrow{x} + \overrightarrow{t},$$

where S is an $m \times d$ matrix and $t \in \mathbb{R}^m$.

Although in optimization theory the terms "affine" and "linear" are usually used interchangeably (recall Definition 2.32), here we distinguish among them. To be specific, "affine" is more general than "linear" because of the $\overrightarrow{t}$ vector (if $\overrightarrow{t} = \overrightarrow{0}$ we recover the linear case).

We can now enumerate some particular instances of constrained optimization problems. The simplest one is:

Definition 2.55 (Linear Program)**.** A *linear program* is a constrained optimization problem where the objective function is linear and all the constraints are affine. Therefore, it can be expressed in the form:

$$\begin{aligned} \min_{\overrightarrow{x}} \quad & \langle \overrightarrow{c}, \overrightarrow{x} \rangle \\ \text{s.t.} \quad & \begin{cases} A\overrightarrow{x} \leq \overrightarrow{b}, \\ E\overrightarrow{x} = \overrightarrow{d}. \end{cases} \end{aligned}$$

In this formulation, each inequality constraint is included in the rows of the $m \times d$ matrix A and its corresponding component b_i. The same happens with the equality constraints and the $p \times d$ matrix E and the vector $\overrightarrow{d}$. As for the objective function f, it becomes a standard dot product in $\mathbb{R}^d$ with a vector $\overrightarrow{c}$.

For the next particular instance, we need this definition:

Definition 2.56 ((Real) Quadratic Function)**.** A *real quadratic function* is a function $f : \mathbb{R}^d \to \mathbb{R}$ that is expressible in the form

$$f\left(\overrightarrow{x}\right) = \frac{1}{2}\left\langle \overrightarrow{x}, Q\overrightarrow{x} \right\rangle + \left\langle \overrightarrow{c}, \overrightarrow{x} \right\rangle,$$

where Q is a symmetric $d \times d$ matrix and $c \in \mathbb{R}^d$.

And now we have:

Definition 2.57 (Quadratic Program)**.** A *quadratic program* is a constrained optimization problem where the objective function is quadratic and all the constraint functions are affine. Therefore, it can be expressed in the form:

$$\begin{aligned} \min_{\overrightarrow{x}} \quad & \frac{1}{2}\left\langle \overrightarrow{x}, Q\overrightarrow{x} \right\rangle + \left\langle \overrightarrow{c}, \overrightarrow{x} \right\rangle \\ \text{s.t.} \quad & \begin{cases} A\overrightarrow{x} \leq \overrightarrow{b}, \\ E\overrightarrow{x} = \overrightarrow{d}. \end{cases} \end{aligned}$$

In the particular case where Q is positive semidefinite (Definition 2.23), the objective function is convex, so that it can be minimized if it is bounded in the feasible region, and this region is non–empty. If Q is positive definite, it is strictly convex, so the minimum is unique if it exists, according to Theorem 2.50. Note that a linear program is a particular case of a quadratic program with Q the null matrix.

As for the final instance:

Definition 2.58 (Convex Program)**.** A *convex program* is a constrained optimization problem where the objective function and the inequality constraint functions are convex and the equality constraint functions are affine. Therefore, it can be expressed in the form:

$$\begin{aligned} \min_{\overrightarrow{x}} \quad & f\left(\overrightarrow{x}\right) \\ \text{s.t.} \quad & \begin{cases} g_i\left(\overrightarrow{x}\right) \leq 0, & i = 1, \ldots, m, \\ E\overrightarrow{x} = \overrightarrow{d}, \end{cases} \end{aligned}$$

where f and $g_i, i = 1, \ldots, m$ are convex.

2.4.3 Duality

Next we move to the subject of how to characterize an optimal solution (that is, a global minimum) for a convex program. Even if this minimum may not be unique, in optimization we are rarely interested in the set of all minima. Most commonly it suffices to obtain a minimum, no matter which one it is.

In order to manage this, we must generalize Lagrangian theory, which was developed by the French mathematician Joseph Louis Lagrange to study optimization problems subject to equality constraints. Here, we extend this treatment to cover also inequality constraints. To begin with, let us define the generalized Lagrangian function:

Definition 2.59 ((Generalized) Lagrangian Function)**.** Given a constrained optimization problem as in Definition 2.53, the *generalized Lagrangian function* (also *Lagrangian function* or simply *Lagrangian*) is the function $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ given by

$$\mathcal{L}\left(\overrightarrow{x}, \overrightarrow{\alpha}, \overrightarrow{\beta}\right) = f\left(\overrightarrow{x}\right) + \sum_{i=1}^{m} \alpha_i g_i\left(\overrightarrow{x}\right) + \sum_{i=1}^{p} \beta_i h_i\left(\overrightarrow{x}\right),$$

where the $\alpha_i, i = 1, \ldots, m$ are the *inequality Lagrange multipliers* and the $\beta_i, i = 1, \ldots, p$ are called *equality Lagrange multipliers*.

From this Lagrangian function we can define the dual function in the following way:

Definition 2.60 (Dual Function)**.** Given a constrained optimization problem as in Definition 2.53 and its corresponding Lagrangian function of Definition 2.59, the *dual function* is a function $\theta : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ defined as

$$\theta\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right) = \inf_{\overrightarrow{x}} \mathcal{L}\left(\overrightarrow{x}, \overrightarrow{\alpha}, \overrightarrow{\beta}\right).$$

The Lagrangian dual problem consists of maximizing the dual function with the constraint that the inequality Lagrange multipliers are positive:

Definition 2.61 (Lagrangian Dual Problem)**.** Given a constrained optimization problem as in Definition 2.53 and its corresponding dual function of Definition 2.59, the *Lagrangian dual problem* is the maximization problem

$$\begin{array}{ll} \max_{\overrightarrow{\alpha}, \overrightarrow{\beta}} & \theta\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right) \\ \text{s.t.} & \alpha_i \geq 0, \quad i = 1, \ldots, m. \end{array}$$

It can be shown that θ is a concave function, even if the optimization problem is not convex. Therefore, the above problem is a convex optimization problem itself, because

maximizing a concave function is equivalent to minimizing a convex one. In contrast to this dual problem, the formulation in Definition 2.53 is usually referred to as *primal problem*. Thus, from now on the term "primal" will be associated to such a problem, whereas the word "dual" will refer to Definition 2.61.

Using the definitions given above, the following theorem is very easy to derive:

Theorem 2.62 (Weak Duality)**.** *Given a feasible point* $\vec{x}$ *for the primal problem and a feasible point* $\left(\vec{\alpha}, \vec{\beta}\right)$ *for the dual problem, it holds that* $\theta\left(\vec{\alpha}, \vec{\beta}\right) \leq f\left(\vec{x}\right)$.

This means that the dual value is upper bounded by the primal value. This also holds for any optimal primal value $\vec{x}^*$ and any optimal dual values $\vec{\alpha}^*$, $\vec{\beta}^*$, so that we have

$$\theta\left(\vec{\alpha}^*, \vec{\beta}^*\right) \leq f\left(\vec{x}^*\right).$$

The difference $f\left(\vec{x}^*\right) - \theta\left(\vec{\alpha}^*, \vec{\beta}^*\right)$ is known as the *duality gap*, which is shown graphically in Figure 2.8.

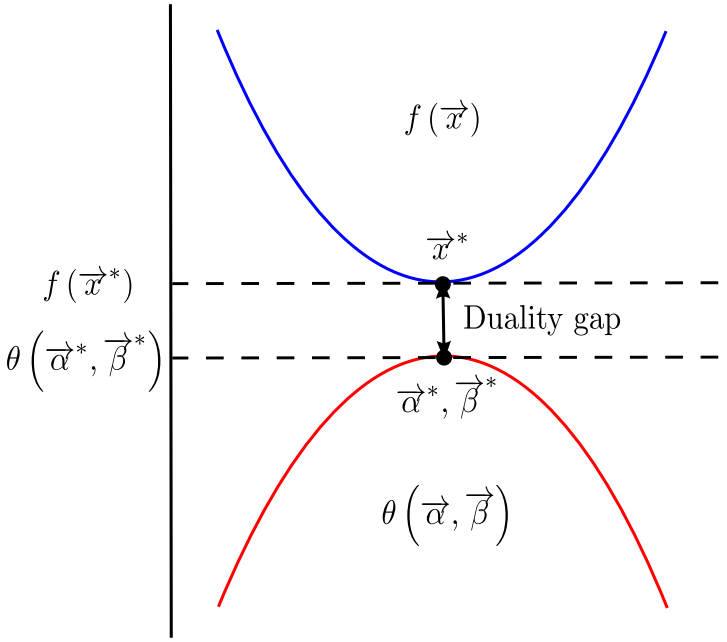

FIGURE 2.8: Illustration of the duality gap, which is the difference between the global minimum of the primal problem, whose objective function is convex, and the global maximum of the dual problem, whose objective function is concave.

We are particularly interested in knowing when this duality gap is 0, so that the primal $f\left(\vec{x}^*\right)$ and dual $\theta\left(\vec{\alpha}^*, \vec{\beta}^*\right)$ optimal values are identical. In order for this to hold, we need some further restrictions [3]:

Theorem 2.63 (Strong Duality)**.** *The duality gap of a convex program with affine equality and inequality constraints is zero.*

Fortunately, the problems we will deal with are all of this kind. The fact that the duality gap is zero is very important, for it opens us two different strategies to solve a given primal problem: either solving it directly or solving its dual. It is usually the case that the dual is simpler to solve. In fact, for SVMs the usual scenario is that the primal problem is convex (and, as we know, with affine constraints) and that the dual problem is quadratic (also with affine constraints). We will also see that the kernel trick plays an important role, so that in practice the dual problems are the ones actually solved.

It remains to see how we can exactly switch from the primal to the dual problem, because the infimum associated to the latter (recall Definition 2.61) is not easily tractable. In most of the cases, this will be simple enough, because the objective function and the constraint functions will all be differentiable. Thus, we can obtain the infimum (in fact, the minimum) by enforcing that the gradient of the Lagrangian be zero, so we will have a dual problem of the form

$$\begin{array}{ll} \max_{\overrightarrow{\alpha},\overrightarrow{\beta}} & \mathcal{D}\left(\overrightarrow{\alpha},\overrightarrow{\beta}\right) \\ \text{s.t.} & \alpha_i \geq 0, \;\; i = 1,\ldots,m, \end{array}$$

where $\mathcal{D}\left(\overrightarrow{\alpha},\overrightarrow{\beta}\right)$ is the function that yields after substitution of the gradient of the Lagrangian set to zero. This should not be confused with the function θ in Definition 2.60, which is defined as a general infimum, not dependent thus on its differentiability.

If either the objective function or the constraint functions are not differentiable, we need to resort to other techniques. For simple cases (fortunately, the problems we will study fall into this category), one possibility is using the *convex conjugate*, which is also expressed in terms of an infimum. This concept is introduced next, after a preliminary definition:

Definition 2.64 ((Real) Dual Space)**.** Given a real vector space V, the *dual space* V^* is the set of all linear maps $f : V \to \mathbb{R}$ (recall Definition 2.32), endowed with the following operations for all $f, f' \in V^*$, $\overrightarrow{x} \in V$ and $\lambda \in \mathbb{R}$:

- $(f + f')\left(\overrightarrow{x}\right) = f\left(\overrightarrow{x}\right) + f'\left(\overrightarrow{x}\right)$ (addition),
- $(\lambda f)\left(\overrightarrow{x}\right) = \lambda f\left(\overrightarrow{x}\right)$ (scalar multiplication).

Notationally we use a star symbol $*$, which should not be confused with the star used for optimal points. The definition of the convex conjugate is:

Definition 2.65 (Convex Conjugate)**.** Given a real normed vector space $\mathcal{X}$ whose dual space is $\mathcal{X}^*$, and a function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ allowed to have infinite values, the *convex conjugate* (also *Fenchel transformation* or *Fenchel dual*) of f is the function $f^* : \mathcal{X}^* \to \mathbb{R}$ defined as:

$$f^*(\overrightarrow{x}^*) = \sup\{\langle \overrightarrow{x}, \overrightarrow{x}^* \rangle - f(\overrightarrow{x}) \,|\, \overrightarrow{x} \in \mathcal{X}\} = -\inf\{f(\overrightarrow{x}) - \langle \overrightarrow{x}, \overrightarrow{x}^* \rangle \,|\, \overrightarrow{x} \in \mathcal{X}\}.$$

That is to say, the convex conjugate of a function on $\mathcal{X}$ is another function (which is also convex) defined in the dual space of $\mathcal{X}$, and the relationship between them is the one stated above.

It also remains to see what conditions indicate that we are at an optimum. These are referred to with the general term of *optimality conditions*. In the particular differentiable case we have described, these optimality conditions are called *Karush–Kuhn–Tucker* (KKT) *conditions*, named after these three researchers [41, 42]. One of these conditions is precisely that the gradient of the Lagrangian is zero, but there are further conditions:

Definition 2.66 (KKT conditions)**.** Given a convex program as in Definition 2.58, with affine differentiable constraints and a differentiable objective function, for a feasible point $\overrightarrow{x}^*$ to be a global minimum, the existence of $\overrightarrow{\alpha}^*$, $\overrightarrow{\beta}^*$ such that the following conditions hold is necessary and sufficient:

- $\frac{\partial \mathcal{L}\left(\overrightarrow{x}^*, \overrightarrow{\alpha}^*, \overrightarrow{\beta}^*\right)}{\partial \overrightarrow{x}^*} = \overrightarrow{0}$ (stationarity),
- $h_i(\overrightarrow{x}^*) = 0$, $i = 1, \ldots, p$ (equality primal feasibility),
- $g_i(\overrightarrow{x}^*) \leq 0$, $i = 1, \ldots, m$ (inequality primal feasibility),
- $\alpha_i^* \geq 0$, $i = 1, \ldots, m$ (dual feasibility),
- $\alpha_i^* g_i(\overrightarrow{x}^*) = 0$, $i = 1, \ldots, m$ (complementary slackness).

These KKT conditions can be thus summarized in an optimal point being: 1) feasible both in the primal in the dual, 2) with the gradient of the Lagrangian equal to zero, 3) with complementary slackness. This last condition means that either an inequality constraint is *active* (that is, $g_i(\overrightarrow{w}^*) = 0$) or its corresponding multiplier is $\alpha_i^* = 0$.

When the objective function or the constraint functions are not differentiable, the KKT conditions remain the same except for the stationarity one, since we cannot calculate the gradient of the Lagrangian function. It turns out that this condition has to be

generalized in a way that implies that the infimum of the Lagrangian is attained [39]. Fortunately, in the next chapters we will see that this KKT stationarity condition will be simple enough to fulfill, either by differentiability or by resorting to the convex conjugate.

Chapter 3

Support Vector Machines

Chapter 1 introduced the concepts of SRM, regularization and linear machines. Later, we learnt in Chapter 2 about kernels and optimization theory. Support Vector Machines merge naturally all these concepts. As a result, a convex problem is formulated that deals with the task at hand, be it classification, regression or unsupervised learning. This is what is known as the primal problem. By means of Lagrangian duality we can derive the corresponding dual problem.

In the following we will examine the different primal and dual formulations for Support Vector Classification, Support Vector Regression and unsupervised learning with One–Class SVMs. Later on, we will consider related geometrical formulations based on finding the closest points in convex hulls.

3.1 Support Vector Classification (SVC)

The literature on SVMs started with the famous paper by Cortes and Vapnik [21], which only dealt with classification tasks. The two previous chapters have provided us with the basic ingredients to understand the formulation these authors suggested. We just need to explore a bit further some of the ideas introduced in those chapters.

3.1.1 Hard–Margin SVC

First of all, we need to recall the notion of margin of Definitions 1.12 and 1.13 and Figure 1.8. Basically, the margin of a point measured the distance of that point to the hyperplane. The margin of a training set was given by the minimum margin of all the points belonging to the training set. After that, we introduced the idea of a maximal

margin hyperplane, which was the hyperplane that had as a result the largest margin for the training set.

The idea of SVC is to find the maximal margin hyperplane for a given training set. To this aim, it is assumed that this hyperplane is *canonical* with respect to that training set. Suppose that the maximal margin hyperplane is given by $\langle \vec{w}^*, \vec{x} \rangle + b^* = 0$, as in Definition 1.14. Being canonical means that $\langle \vec{w}^*, \vec{x}^*_+ \rangle + b^* = +1$, where $\vec{x}^*_+$ is the point of the positive class with minimal margin. Conversely, $\langle \vec{w}^*, \vec{x}^*_- \rangle + b^* = -1$, with $\vec{x}^*_-$ the point of the negative class with minimal margin.

With such a canonical hyperplane, it follows that $\langle \vec{w}^*, \vec{x}^*_+ - \vec{x}^*_- \rangle = 2$, because the bias terms cancel. If we also recall that normalizing the weight vector implied measuring the actual distances (what was called geometric margin), this means that the distance between $\vec{x}^*_+$ and $\vec{x}^*_-$ is given by $\left\langle \frac{\vec{w}^*}{\|\vec{w}^*\|_2}, \vec{x}^*_+ - \vec{x}^*_- \right\rangle = \frac{2}{\|\vec{w}^*\|_2}$. The hyperplanes $\langle \vec{w}^*, \vec{x}_i \rangle + b^* = \pm 1$ are called *support hyperplanes*. All this is illustrated in Figure 3.1.

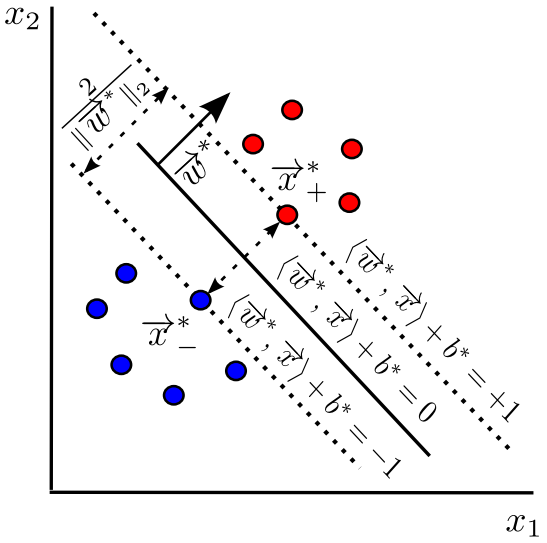

FIGURE 3.1: Illustration of the canonical margin (i.e. the distance between both support hyperplanes, printed with dashed lines) obtained in hard–margin SVC. The optimal hyperplane is printed with a continuous line.

It is then clear that, in order to find the maximal margin hyperplane we can formulate the following problem:

$$\begin{aligned} \min_{\vec{w},b} \quad & \frac{1}{2}\|\vec{w}\|_2^2 \\ \text{s.t.} \quad & y_i \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \geq 1, \; i = 1, \ldots, N. \end{aligned} \tag{3.1}$$

The objective function maximizes the distance $2/\|\overrightarrow{w}\|_2$ by minimizing $\|\overrightarrow{w}\|_2$. Ignoring the factor of 2, this is obviously equivalent. Squaring the norm has no effect, because if we minimize $\|\overrightarrow{w}\|_2^2$ we will also minimize $\|\overrightarrow{w}\|_2$. However, it is convenient to keep the square, because then the function is more easily differentiable.

Regarding the constraints, they summarize the requirement of canonical hyperplanes: the points must lie at least at a distance 1 from the hyperplane. Moreover, perfect classification is assumed (for the time being, the classes are assumed to be linearly separable), so that $\langle\overrightarrow{w}, \overrightarrow{x}_i\rangle + b \geq +1$ for the positive class and $\langle\overrightarrow{w}, \overrightarrow{x}_i\rangle + b \leq -1$ for the negative class.

Thus, the solution $(\overrightarrow{w}^*, b^*)$ to the above problem yields the maximal margin hyperplane. Problem (3.1) is known as the primal problem for *hard–margin SVC*. Here, the word "hard" emphasizes that we are assuming a linearly separable sample, so that we can find a canonical hyperplane satisfying the constraints. In the real world, this is seldom the case, and a more flexible formulation is needed, as will be seen in the following sections.

Let us go back to (3.1). Recalling Definition 2.58, it is clearly a convex program: the objective function is convex, and so are the inequality constraints. It is simply a matter of rearrangement to write (3.1) in the standard form of Definition 2.53, that is, with inequality constraints of the form $\leq$ instead of $\geq$.

Therefore, we can apply the results given in Subsection 2.4.3 to derive the corresponding dual problem. The Lagrangian (recall Definition 2.59) is given by

$$\mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\alpha}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i \left[y_i(\langle\overrightarrow{w}, \overrightarrow{x}_i\rangle + b) - 1\right]. \tag{3.2}$$

According to Definitions 2.60 and 2.61, the dual problem consists of maximizing the infimum of (3.2) with respect to the primal variables $\overrightarrow{w}$ and b, subject to the non–negativity of the $\overrightarrow{\alpha}$ coefficients. Since (3.2) is differentiable, this infimum is implicitly found by requiring that the gradient of the Lagrangian be zero, which means in this case

$$\begin{array}{rcl} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} = \overrightarrow{w} - \sum_{i=1}^{N} \alpha_i y_i \overrightarrow{x}_i = 0 & \Rightarrow & \overrightarrow{w} = \sum_{i=1}^{N} \alpha_i y_i \overrightarrow{x}_i, \\ \frac{\partial\mathcal{L}}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 & \Rightarrow & \sum_{i=1}^{N} \alpha_i y_i = 0. \end{array} \tag{3.3}$$

Special attention must be given to the first equation. It establishes that the weight vector, which is normal to the hyperplane, can be written as a combination of the different points of the sample. In fact, the only points that influence on $\overrightarrow{w}$ are those for which the associated Lagrange multipliers are positive. This subset of points of the training set are called *support vectors*, hence the name of Support Vector Machines.

Substituting (3.3) in (3.2) yields the dual function (cf. Definition 2.60):

$$\theta\left(\overrightarrow{\alpha}\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left\langle \overrightarrow{x}_i, \overrightarrow{x}_j \right\rangle - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left\langle \overrightarrow{x}_i, \overrightarrow{x}_j \right\rangle + \sum_{i=1}^{N}\alpha_i,$$

so that the dual problem of the primal (3.1) is

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left\langle \overrightarrow{x}_i, \overrightarrow{x}_j \right\rangle - \sum_{i=1}^{N}\alpha_i \\ \text{s.t.} \quad & \begin{cases} \alpha_i \geq 0, \;\; i = 1, \ldots, N, \\ \sum_{i=1}^{N}\alpha_i y_i = 0. \end{cases} \end{aligned} \tag{3.4}$$

Note that we have switched the max in Definition 2.61 to a min, so as to minimize a convex function, analogously to what the primal does. If we choose to solve this dual formulation, obtaining a solution $\overrightarrow{\alpha}^*$, the primal solution can be easily recovered: by (3.3) we have that $\overrightarrow{w}^* = \sum_{i=1}^{N}\alpha_i^* y_i \overrightarrow{x}_i = \sum_S \alpha_i^* y_i \overrightarrow{x}_i$, where S stands for the indices of the support vectors (SVs), which have $\alpha_i^* > 0$.

The bias b^* is somewhat more difficult to recover, because we need to resort to the KKT optimality conditions (cf. Definition 2.66). The conditions establish for this case that $(\overrightarrow{w}^*, b^*)$ is an optimal solution of (3.1) if and only if there exists an $\overrightarrow{\alpha}^*$ for (3.4) such that:

- $\overrightarrow{w}^* = \sum_{i=1}^{N}\alpha_i^* y_i \overrightarrow{x}_i$ (stationarity 1),
- $\sum_{i=1}^{N}\alpha_i^* y_i = 0$ (stationarity 2),
- $y_i(\langle \overrightarrow{w}^*, \overrightarrow{x}_i \rangle + b^*) \geq 1, \; i = 1, \ldots, N$ (inequality primal feasibility),
- $\alpha_i^* \geq 0, \; i = 1, \ldots, N$ (dual feasibility),
- $\alpha_i^* \left[y_i(\langle \overrightarrow{w}^*, \overrightarrow{x}_i \rangle + b^*) - 1\right] = 0, \; i = 1, \ldots, N$ (complementary slackness).

The complementary slackness condition means that $y_i(\langle \overrightarrow{w}^*, \overrightarrow{x}_i \rangle + b^*) - 1 = 0$ for every $i \in S$, so that $\langle \overrightarrow{w}^*, \overrightarrow{x}_i \rangle + b^* = y_i$ for any support vector, and thus we can calculate $b^* = y_i - \langle \overrightarrow{w}^*, \overrightarrow{x}_i \rangle$. Nevertheless, for the sake of numerical stability, it is usually recommended [3] that this bias be averaged as

$$b^* = \frac{1}{|S|}\sum_{S}\left(y_i - \langle \overrightarrow{w}^*, \overrightarrow{x}_i \rangle\right). \tag{3.5}$$

Once we have $\overrightarrow{w}^*$ and b^*, and referring back to Equation 1.9, the decision function learnt by the Hard–Margin SVM is

$$f^*\left(\overrightarrow{x}\right) = \langle \overrightarrow{w}^*, \overrightarrow{x} \rangle + b^* = \sum_S \alpha_i^* y_i \left\langle \overrightarrow{x}_i, \overrightarrow{x} \right\rangle + b^*. \tag{3.6}$$

Observe that in this decision function, as well as in the primal (3.1) and the dual (3.4), the only operations among vectors are inner products. Therefore, if we introduce a kernel function (for instance a kernel from §2.2.4) we can effectively learn a linear machine in the feature space. This increases substantially the expressive power of SVMs, because in the input space the decision function (3.6) translates to a non–linear decision boundary, as was illustrated in Figure 1.11.

In order for this kernelization to be possible, we need to change (3.1) to the following:

$$\begin{aligned} \min_{\overrightarrow{w},b} \quad & \frac{1}{2}\|\overrightarrow{w}_{\mathcal{H}}\|_2^2 \\ \text{s.t.} \quad & y_i\left(\langle \overrightarrow{w}_{\mathcal{H}}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) \geq 1, \quad i = 1, \ldots, N, \end{aligned} \tag{3.7}$$

where patterns $\overrightarrow{x}_i$ are now substituted by their projections $\Phi\left(\overrightarrow{x}_i\right)$ onto the feature space $\mathcal{H}$ with the feature map $\Phi\left(\cdot\right)$. As was described in §1.3.2, this change implies that $\overrightarrow{w}$ now also lies in $\mathcal{H}$, hence the explicit subscript $\mathcal{H}$.

Recall however that we do not care about what this feature space $\mathcal{H}$ is, be it the RKHS of §2.2.2, the Mercer space of §2.2.3 or some other space that satisfies $k\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) = \left\langle \Phi\left(\overrightarrow{x}_i\right), \Phi\left(\overrightarrow{x}_j\right)\right\rangle$, for the chosen kernel k. It might seem that we cannot solve this problem, because we do not know which is the feature map $\Phi\left(\cdot\right)$.

That is indeed the case for the primal problem, but fortunately the dual is still solvable by using kernels:

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K_{ij} - \sum_{i=1}^N \alpha_i \\ \text{s.t.} \quad & \begin{cases} \alpha_i \geq 0, & i = 1, \ldots, N, \\ \sum_{i=1}^N \alpha_i y_i = 0, \end{cases} \end{aligned} \tag{3.8}$$

where K is a kernel matrix, as in Definition 2.22. The above result is obtained because now the weight vector becomes

$$\overrightarrow{w}_{\mathcal{H}} = \sum_{i=1}^{N} \alpha_i y_i \Phi\left(\overrightarrow{x}_i\right),$$

which in turn implies that the optimal decision function is now

$$f^*\left(\overrightarrow{x}\right) = \langle \overrightarrow{w}^*_{\mathcal{H}}, \Phi\left(\overrightarrow{x}\right)\rangle + b^* = \sum_{S} \alpha_i^* y_i k\left(\overrightarrow{x}_i, \overrightarrow{x}\right) + b^*, \tag{3.9}$$

where we used $k\left(\overrightarrow{x}_i, \overrightarrow{x}\right) = \langle \Phi\left(\overrightarrow{x}_i\right), \Phi\left(\overrightarrow{x}\right)\rangle$. The KKT conditions and the calculation of b^* are derived analogously to the non–kernelized case.

Note that this non–kernelized case can also be seen as a kernelized case with the linear kernel $K_{ij} = \langle \overrightarrow{x}_i, \overrightarrow{x}_j \rangle$. This is why it is usually referred to as "linear SVM".

We see then that working in the feature space in SVMs is transparent by means of kernel functions. From now on, we will always assume that kernels are used (in the SVM jargon, we assume a *kernel setting*), working thus with the projections $\Phi\left(\overrightarrow{x}_i\right)$ in the feature space. In order to keep the notation uncluttered, we will write $\overrightarrow{w}$ instead of $\overrightarrow{w}_{\mathcal{H}}$, although it should be kept in mind that this vector also lies in the feature space $\mathcal{H}$.

3.1.2 l_2–SVC

There are still some issues that needed to be solved after the publication of [21] to make SVMs practical. The main issue is that, even if we implicitly project the patterns to a high–dimensional feature space through a kernel, it might be the case that the patterns are still linearly inseparable. If this happens, problem (3.7) will be infeasible.

Clearly we need to allow for potential misclassifications. If we recall Definition 1.15 and Figure 1.9 for margin slack variables, these will help us to quantify the number and magnitude of misclassifications. Because we are dealing with canonical hyperplanes, we will have $\gamma = +1$ in Figure 1.9.

Thus, the slack variables in SVC are equal to

$$\xi_i = \max\left\{0, 1 - y_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right)\right\}. \tag{3.10}$$

At this point we have two conflicting goals: on the one hand we want to maximize the margin, and on the other hand we would like to minimize the number and magnitude of misclassifications. Assuming that the sample is linearly inseparable, the wider

the margin, the larger previous misclassifications become. This brings us to mind the regularization framework of §1.2.4, where a penalty term C was used to penalize model complexity, acting as a trade–off between the complexity term and the ERM term. Here, the complexity term is the norm of $\overrightarrow{w}$.

l_2–SVC suggests to penalize with the trade–off parameter C the sum of the squares of the slack variables (3.10). Recalling (1.3), this corresponds to using a squared loss function. The primal problem becomes then:

$$\begin{aligned} \min_{\overrightarrow{w},b,\overrightarrow{\xi}} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 \\ \text{s.t.} \quad & y_i\left(\langle\overrightarrow{w},\Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) \geq 1-\xi_i, \quad i=1,\ldots,N. \end{aligned} \tag{3.11}$$

One may ask why this problem does not take into account the fact that $\xi_i \geq 0$ in Equation (3.10). A simple observation clarifies that adding these constraints is unnecessary: if $y_i(\langle\overrightarrow{w},\Phi\left(\overrightarrow{x}_i\right)\rangle + b) > 1$ we will not take $\xi_i < 0$, but $\xi_i = 0$, because this choice minimizes the objective function (C is always assumed to be positive, so the term $\frac{C}{2}\sum_{i=1}^{N}\xi_i^2$ is non–negative). Conversely, if $y_i(\langle\overrightarrow{w},\Phi\left(\overrightarrow{x}_i\right)\rangle + b) \leq 1$ we take then the smallest ξ_i possible, which turns out to be $\xi_i = 1 - y_i(\langle\overrightarrow{w},\Phi\left(\overrightarrow{x}_i\right)\rangle + b)$. Thus, we are fulfilling (3.10).

The first term of the objective function deals with margin maximization, whereas the second term is intended to keep misclassifications at bay. Observe the analogy with Equation (1.8). Depending on the specific value of C, we will tend to maximize the margin (small C) or to minimize misclassifications (large C). In fact, in the limit case of $C = +\infty$, problem (3.11) becomes (3.7), because the slack variables are penalized infinitely, and they tend to zero to compensate for this penalty.

This last fact becomes more evident in the dual. The Lagrangian for (3.11) is

$$\mathcal{L}\left(\overrightarrow{w},b,\overrightarrow{\xi},\overrightarrow{\alpha}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i\left[y_i(\langle\overrightarrow{w},\Phi\left(\overrightarrow{x}_i\right)\rangle + b) - 1 + \xi_i\right]. \tag{3.12}$$

Requiring that its gradient be zero implies

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} = \overrightarrow{w} - \textstyle\sum_{i=1}^{N}\alpha_i y_i\Phi\left(\overrightarrow{x}_i\right) = 0 \quad &\Rightarrow \quad \overrightarrow{w} = \textstyle\sum_{i=1}^{N}\alpha_i y_i\Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial\mathcal{L}}{\partial b} = -\textstyle\sum_{i=1}^{N}\alpha_i y_i = 0 \quad &\Rightarrow \quad \textstyle\sum_{i=1}^{N}\alpha_i y_i = 0, \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = C\xi_i - \alpha_i = 0 \quad &\Rightarrow \quad \alpha_i = C\xi_i. \end{aligned} \tag{3.13}$$

Substituting the above in (3.12) produces the dual function

$$\theta\left(\overrightarrow{\alpha}\right) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K_{ij} - \frac{1}{2C} \sum_{i=1}^{N} \alpha_i^2.$$

Rearranging terms, and writing the dual as a minimization problem yields

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left(K_{ij} + \frac{\delta_{ij}}{C} \right) - \sum_{i=1}^{N} \alpha_i \\ s.t. \quad & \begin{cases} 0 \leq \alpha_i, & i = 1, \ldots, N, \\ \sum_{i=1}^{N} \alpha_i y_i = 0, \end{cases} \end{aligned} \tag{3.14}$$

where δ_{ij} stands for Kronecker's delta symbol, with $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Note that this dual is identical to (3.8) except for the extra term δ_{ij}/C in the objective function. When $C = +\infty$ this term becomes zero, recovering the hard–margin case.

In fact, it can also be considered in the following way [43]: l_2–SVC and hard–margin SVC are equivalent problems, once we change the kernel function in l_2–SVC to

$$\tilde{k}\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) = k\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) + \frac{\delta_{ij}}{C}. \tag{3.15}$$

For this to hold, we need to specify a new feature map $\tilde{\Phi}\left(\cdot\right)$ such that $\tilde{k}\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) = \left\langle \tilde{\Phi}\left(\overrightarrow{x}_i\right), \tilde{\Phi}\left(\overrightarrow{x}_j\right) \right\rangle$. This can be done through

$$\tilde{\Phi}\left(\overrightarrow{x}_i\right) = \left(\Phi\left(\overrightarrow{x}_i\right), \frac{y_i}{\sqrt{C}} \overrightarrow{e}_i \right), \tag{3.16}$$

where $\overrightarrow{e}_i$ stands for an N–dimensional vector with its i–th component set to 1 and the rest set to 0. Thus, the second vector in (3.16) has a value of $y_i/\sqrt{C}$ in its i–th component, and zeros elsewhere.

As for the primal, in order to recover the hard–margin formulation the weight vector is redefined as

$$\overrightarrow{w} = \left(\overrightarrow{w}, \sqrt{C}\, \overrightarrow{\xi} \right), \tag{3.17}$$

so that $(1/2) \left\| \overrightarrow{w} \right\|_2^2 = (1/2) \left\| \overrightarrow{w} \right\|_2^2 + (C/2) \sum_{i=1}^{N} \xi_i^2$, and then

$$\left\langle \overrightarrow{w}, \tilde{\Phi}\left(\overrightarrow{x}_i\right) \right\rangle = \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right) \rangle + y_i \xi_i = \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right) \rangle + y_i \frac{\alpha_i}{C}, \tag{3.18}$$

thus transforming (3.11) into (3.7).

In summary, l_2–SVC can be regarded as Hard–Margin SVC in an augmented feature space with associated feature map $\tilde{\Phi}(\cdot)$. As we will see with subsequent l_2 formulations, this is a general result.

Regarding the KKT conditions, they are also in correspondence to the ones for hard–margin SVMs:

$$\begin{aligned} \overrightarrow{w}^* &= \textstyle\sum_{i=1}^N \alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right), \\ \textstyle\sum_{i=1}^N \alpha_i^* y_i &= 0, \\ \alpha_i^* &\geq 0, \qquad i = 1, \ldots, N, \\ y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* \right) &\geq 1 - \tfrac{\alpha_i^*}{C}, \qquad i = 1, \ldots, N \\ \alpha_i^* \left[y_i (\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^*) - 1 + \tfrac{\alpha_i^*}{C} \right] &= 0, \qquad i = 1, \ldots, N. \end{aligned} \tag{3.19}$$

Splitting them in two different cases we get:

- $\alpha_i^* = 0$: then $y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* \right) \geq 1$.
- $\alpha_i^* > 0$: $y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* \right) = 1 - \frac{\alpha_i^*}{C} < 1$.

Since the only equalities are obtained in the second case, that is the one used for finding the optimal bias b^*. As in the hard–margin case, instead of deriving it from a single point, it is preferred to average the result:

$$b^* = \frac{1}{|S'|} \sum_{S'} \left(y_i \left(1 - \frac{\alpha_i^*}{C} \right) - \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle \right), \tag{3.20}$$

where S' is the set of indices of SVs. The final decision function has exactly the same form as in (3.9), so it is not repeated here.

The conditions above establish that the SVs are the points which lie in the wrong side of the support hyperplane of the corresponding class. Note that this also includes the points misclassified by the model. The ones that lie exactly in the support hyperplane are not SVs. At first sight, it may seem surprising that the points that influence in the model are the "worst classified" ones. However, we have to take into account that those are precisely the most informative points (they are "unusual" points, informally

speaking), whereas the points correctly classified are "typical", non–informative points, so they are assigned a zero coefficient.

It is also worth observing that in the augmented feature space the points become linearly separable, and we recover the KKT conditions for the hard–margin case, that is:

- $\alpha_i^* = 0$: then $y_i \left(\left\langle \overrightarrow{\tilde{w}}^*, \tilde{\Phi}\left(\overrightarrow{x}_i\right) \right\rangle + \tilde{b}^* \right) \geq 1$.
- $\alpha_i^* > 0$: $y_i \left(\left\langle \overrightarrow{\tilde{w}}^*, \tilde{\Phi}\left(\overrightarrow{x}_i\right) \right\rangle + \tilde{b}^* \right) = 1$.

Figure 3.2 helps to understand this. Observe the change in the distribution of points and the position of the hyperplanes (of course, the augmented feature space should have more dimensions than the feature space, but we depict both as bidimensional). What is interesting is that the coefficients α_i^* are the same in both cases.

A potential problem of l_2–SVMs is that (3.13) implies that the coefficients α_i^* are directly proportional to the ξ_i^* values. This means that, the "worse" classified a point is, the more influence it will have in the model. It may be the case that *outliers* (points that are abnormally far from the rest of points of that class) influence considerably in the final model, which is not desirable nor robust. To tackle this problem, l_1–SVMs are the standard choice nowadays.

3.1.3 l_1–SVC

l_1–SVC suggests to penalize with the trade–off parameter C the sum of the slack variables (3.10) themselves, instead of using their squares (that is, using a hinge loss instead of a square loss in (1.3)), so that the primal problem becomes

$$\begin{aligned} \min_{\overrightarrow{w}, b, \overrightarrow{\xi}} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \begin{cases} y_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) \geq 1 - \xi_i, & i = 1, \ldots, N, \\ \xi_i \geq 0, & i = 1, \ldots, N. \end{cases} \end{aligned} \tag{3.21}$$

Note how the problem above applies again (3.10): if $y_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) \geq 1$ the optimal choice is $\xi_i = 0$, because the term $C\sum_{i=1}^{N}\xi_i$ is non–negative. Conversely, if $y_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) < 1$ the optimal choice is $\xi_i = 1 - y_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) > 0$.

Because the ξ_i are non–negative, taking the absolute value of them does not change the solution, and this is why it is referred to as l_1, in contrast to an l_2–SVM, which uses

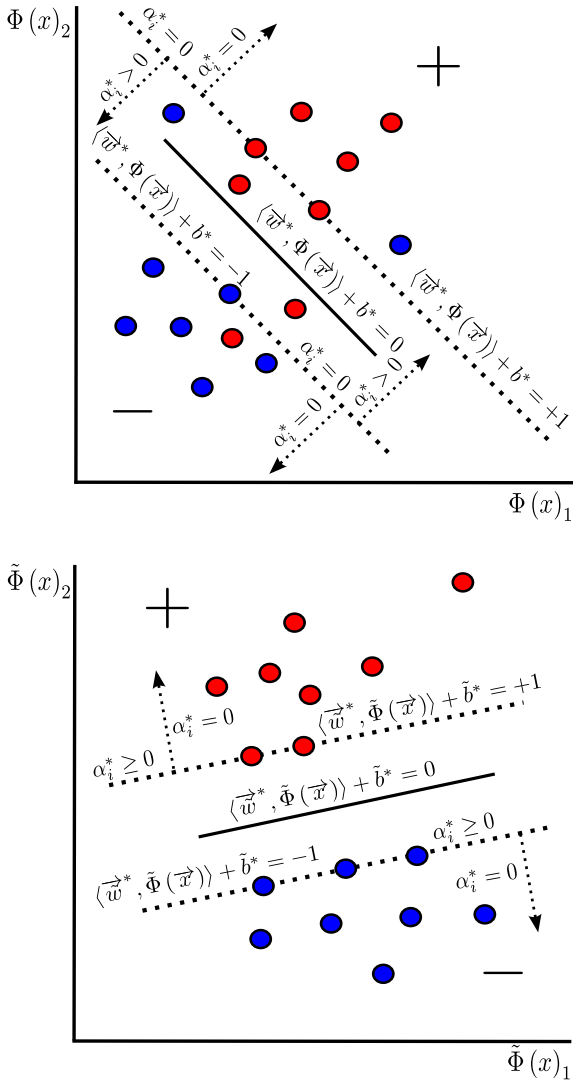

FIGURE 3.2: Depiction of the values of the optimal α_i for l_2–SVC, both in the feature space and in the augmented feature space. Whereas in the feature space the SVs are the points in the wrong side of the corresponding support hyperplane, in the augmented one they must lie in that support hyperplane.

the l_2 (squared) norm of the vector $\overrightarrow{\xi}$ (recall the definition of the l_1 and l_2–norms in Definition 2.30). Both kinds of SVMs are grouped under the name *Soft–Margin SVC*, in contrast to the Hard–Margin case of §3.1.1.

Problem (3.21) also reduces to the hard–margin case when $C = +\infty$. The Lagrangian is

$$\mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \overrightarrow{\alpha}, \overrightarrow{\gamma}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b) - 1 + \xi_i\right] - \sum_{i=1}^{N}\gamma_i\xi_i, \tag{3.22}$$

where we introduce additional Lagrange multipliers γ_i for the constraints $\xi_i \geq 0$. Requiring that its gradient be zero implies now

$$\begin{array}{rcl} \frac{\partial \mathcal{L}}{\partial \overrightarrow{w}} = \overrightarrow{w} - \sum_{i=1}^{N} \alpha_i y_i \Phi\left(\overrightarrow{x}_i\right) = 0 & \Rightarrow & \overrightarrow{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 & \Rightarrow & \sum_{i=1}^{N} \alpha_i y_i = 0, \\ \frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \gamma_i = 0 & \Rightarrow & 0 \leq \alpha_i \leq C. \end{array} \tag{3.23}$$

The last inequality stems from the fact that $\gamma_i \geq 0$ and $\alpha_i \geq 0$ for being inequality Lagrange multipliers. Substituting the above in (3.22) produces the dual

$$\begin{array}{ll} \min\limits_{\overrightarrow{\alpha}} & \frac{1}{2} \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{N} \alpha_i \alpha_j y_i y_j K_{ij} - \sum\limits_{i=1}^{N} \alpha_i \\ \text{s.t.} & \begin{cases} 0 \leq \alpha_i \leq C, & i = 1, \ldots, N, \\ \sum_{i=1}^{N} \alpha_i y_i = 0. \end{cases} \end{array} \tag{3.24}$$

Note that this dual is identical to (3.8) except for the upper–bound C in the α_i multipliers. When $C = +\infty$ this upper–bound does not have any effect, recovering the hard–margin case.

Regarding the KKT conditions, these are somewhat more complex than for hard–margin or l_2–SVMs:

- $\overrightarrow{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right)$ (stationarity 1),
- $\sum_{i=1}^{N} \alpha_i^* y_i = 0$ (stationarity 2),
- $\alpha_i^* = C - \gamma_i^*,\ i = 1, \ldots, N$ (stationarity 3),
- $y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^*\right) \geq 1 - \xi_i^*,\ i = 1, \ldots, N$ (inequality primal feasibility 1),
- $\xi_i^* \geq 0,\ i = 1, \ldots, N$ (inequality primal feasibility 2),
- $\alpha_i^* \geq 0,\ i = 1, \ldots, N$ (dual feasibility 1),
- $\gamma_i^* \geq 0,\ i = 1, \ldots, N$ (dual feasibility 2),
- $\alpha_i^* \left[y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^*\right) - 1 + \xi_i^*\right] = 0,\ i = 1, \ldots, N$ (complementary slackness 1),
- $\gamma_i^* \xi_i^* = 0,\ i = 1, \ldots, N$ (complementary slackness 2).

Eliminating $\overrightarrow{\gamma}^*$, these conditions can be rewritten more compactly as:

$$\begin{aligned}
\overrightarrow{w}^* &= \textstyle\sum_{i=1}^N \alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right), \\
\textstyle\sum_{i=1}^N \alpha_i^* y_i &= 0, \\
0 \leq \alpha_i^* &\leq C, \qquad i = 1, \ldots, N, \\
y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &\geq 1 - \xi_i^*, \qquad i = 1, \ldots, N, \\
\xi_i^* &\geq 0, \qquad i = 1, \ldots, N, \\
\alpha_i^* \left[y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) - 1 + \xi_i^*\right] &= 0, \qquad i = 1, \ldots, N, \\
\left(C - \alpha_i^*\right) \xi_i^* &= 0, \qquad i = 1, \ldots, N.
\end{aligned} \tag{3.25}$$

Separating them in three different cases, we get

- $\alpha_i^* = 0$: then $\xi_i^* = 0$, so $y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) \geq 1$.
- $0 < \alpha_i^* < C$: then $\xi_i^* = 0$ and $y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) = 1$.
- $\alpha_i^* = C$: then $y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) = 1 - \xi_i^* \leq 1$.

Since the only equalities are obtained in the second case, that is the one used for finding the optimal bias b^*. As in the previous SVMs, instead of deriving it from a single point with $0 < \alpha_i^* < C$, it is preferred to average the result:

$$b^* = \frac{1}{|S'|} \sum_{S'} \left(y_i - \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle\right), \tag{3.26}$$

where S' is the set of indices $\{i : 0 < \alpha_i^* < C\}$. The final decision function has again the same form than in (3.9).

Looking back at the cases above, points with $y_i(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*) > 1$ cannot be SVs. Points with $y_i(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*) < 1$ are always SVs with $\alpha_i^* = C$, whereas points with $y_i(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*) = 1$ (that is, the ones lying in the support hyperplanes) can have any value of α_i^*: 0, C, or something in between.

The term "support vector" might seem a bit controversial here. If this is understood as a point that lies exactly in the support hyperplane, then only points with $0 < \alpha_i^* < C$ are true support vectors, whereas points with $\alpha_i^* = C$ are not. However, in the literature by "support vector" it is usually meant a point with a non–zero coefficient, that is, a point that influences in the model, even if geometrically it does not lie in the corresponding support hyperplane. We will follow this convention in the sequel.

The analysis above is shown graphically in Figure 3.3. Now there is no further interpretation in terms of an augmented feature space. Besides, the robustness issue has been

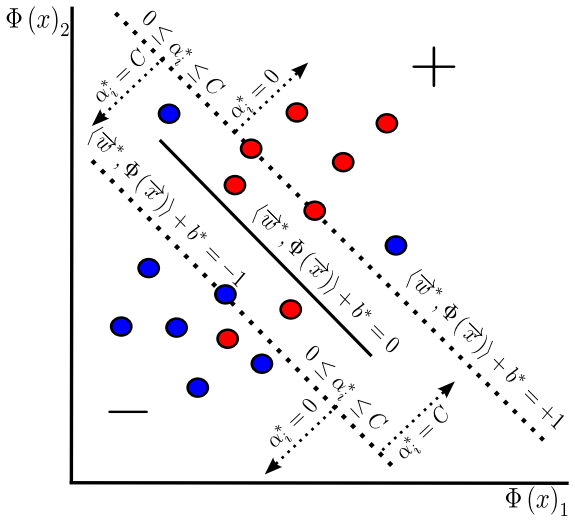

FIGURE 3.3: Depiction of the values of the optimal α_i for l_1–SVC. Points placed at the correct side of their corresponding support hyperplanes have a value of 0, whereas points placed at the wrong side have a value of C. Any value is possible for points lying exactly on the support hyperplanes.

tackled by limiting the influence of a single point to C, whereas it had no bound in the l_2 variant.

3.1.4 ν–SVC

At this point, we might argue that l_1–SVC is a sound and reliable paradigm for classification. That is the case indeed, but it is not straightforward at all to know in advance which is the best value of C for a given problem. As we have already seen, the larger we make it, the more intolerant to misclassifications we are (which tends to overfitting), whereas the smaller we make it, the more we tend to underfit the model. A lot of research has been carried out to automatically tune the C parameter (usually together with the kernel parameters) [44–46], but this is still a difficult issue, and usually *cross–validation* needs to be performed.

An attempt to alleviate this problem which took another direction was the conception of ν–SVMs [23]. In the case of classification (ν–SVC), the idea is to replace the regularization parameter C with another parameter ν, which is in principle easier to tune

because it lies in the interval $(0, 1]$. The primal formulation was originally set as

$$\begin{aligned} \min_{\overrightarrow{w},b,\rho,\overrightarrow{\xi}} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \begin{cases} y_i\left(\langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) \geq \rho - \xi_i, & i = 1, \ldots, N, \\ \xi_i \geq 0, & i = 1, \ldots, N, \\ \rho \geq 0. \end{cases} \end{aligned} \tag{3.27}$$

As stated above, the C parameter has disappeared, but there is instead a new primal variable ρ, which is related to the width of the margin (note that we no longer assume canonical hyperplanes, so ± 1 is replaced by this ρ). However, it was noted in [47] that the constraint $\rho \geq 0$ is redundant. Since $b = \rho = 0$ and $\overrightarrow{w} = \overrightarrow{\xi} = \overrightarrow{0}$ is a feasible solution, any non–trivial solution should give a negative value for the objective function. Besides, if $\nu\rho = 0$, that is indeed the optimal solution, because the other two terms in the objective function are non–negative. It is readily seen then that $\nu\rho > 0$ if we want a meaningful model, so that $\nu > 0$ and $\rho > 0$ as well.

With that constraint removed, the Lagrangian of (3.27) is

$$\begin{aligned} \mathcal{L}\left(\overrightarrow{w}, b, \rho, \overrightarrow{\xi}, \overrightarrow{\alpha}, \overrightarrow{\gamma}\right) = & \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i \\ & - \sum_{i=1}^{N}\alpha_i\left[y_i(\langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b) - \rho + \xi_i\right] - \sum_{i=1}^{N}\gamma_i\xi_i. \end{aligned} \tag{3.28}$$

Equaling the gradient to zero produces

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} = \overrightarrow{w} - \textstyle\sum_{i=1}^{N}\alpha_i y_i \Phi\left(\overrightarrow{x}_i\right) = 0 \quad &\Rightarrow \quad \overrightarrow{w} = \textstyle\sum_{i=1}^{N}\alpha_i y_i \Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial\mathcal{L}}{\partial b} = -\textstyle\sum_{i=1}^{N}\alpha_i y_i = 0 \quad &\Rightarrow \quad \textstyle\sum_{i=1}^{N}\alpha_i y_i = 0, \\ \frac{\partial\mathcal{L}}{\partial\rho} = -\nu + \textstyle\sum_{i=1}^{N}\alpha_i = 0 \quad &\Rightarrow \quad \textstyle\sum_{i=1}^{N}\alpha_i = \nu, \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = \frac{1}{N} - \alpha_i - \gamma_i = 0 \quad &\Rightarrow \quad 0 \leq \alpha_i \leq \frac{1}{N}, \end{aligned} \tag{3.29}$$

giving the dual

$$\begin{array}{ll} \min\limits_{\overrightarrow{\alpha}} & \frac{1}{2}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\alpha_i\alpha_j y_i y_j K_{ij} \\ \text{s.t.} & \begin{cases} 0 \leq \alpha_i \leq \frac{1}{N}, & i = 1,\ldots,N, \\ \sum_{i=1}^{N}\alpha_i y_i = 0, \\ \sum_{i=1}^{N}\alpha_i = \nu, \end{cases} \end{array} \tag{3.30}$$

which is quite similar to (3.24). The influence of any single point is also limited (this time by $1/N$ instead of C). The dual objective function now only has the quadratic term, but there is one further constraint $\sum_{i=1}^{N}\alpha_i = \nu$.

The KKT conditions can be seen to be

$$\begin{aligned} \overrightarrow{w}^* = \textstyle\sum_{i=1}^{N}\alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right), & \\ \textstyle\sum_{i=1}^{N}\alpha_i^* y_i = 0, & \\ \textstyle\sum_{i=1}^{N}\alpha_i^* = \nu, & \\ 0 \leq \alpha_i^* \leq \tfrac{1}{N}, & \quad i = 1,\ldots,N, \\ y_i\left(\langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) \geq \rho^* - \xi_i^*, & \quad i = 1,\ldots,N, \\ \xi_i^* \geq 0, & \quad i = 1,\ldots,N, \\ \alpha_i^*\left[y_i\left(\langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) - \rho^* + \xi_i^*\right] = 0, & \quad i = 1,\ldots,N, \\ \left(\tfrac{1}{N} - \alpha_i^*\right)\xi_i^* = 0, & \quad i = 1,\ldots,N. \end{aligned} \tag{3.31}$$

Similarly to l_1–SVC, we can recover the values ρ^* and b^* from those points such that $0 < \alpha_i^* < \frac{1}{N}$, which imply $y_i\left(\langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) = \rho^*$. Averaging the results for the sake of numerical stability, it is easy to arrive to

$$\begin{aligned} b^* &= -\tfrac{1}{2}\left(\tfrac{1}{|S'_+|}\textstyle\sum_{i\in S'_+}\sum_{j=1}^{N}\alpha_j y_j K_{ij} + \tfrac{1}{|S'_-|}\sum_{i\in S'_-}\sum_{j=1}^{N}\alpha_j y_j K_{ij}\right) \\ \rho^* &= \tfrac{1}{2}\left(\tfrac{1}{|S'_+|}\textstyle\sum_{i\in S'_+}\sum_{j=1}^{N}\alpha_j y_j K_{ij} - \tfrac{1}{|S'_-|}\sum_{i\in S'_-}\sum_{j=1}^{N}\alpha_j y_j K_{ij}\right), \end{aligned} \tag{3.32}$$

where $S'_\pm$ is the set of indices $\left\{i : 0 < \alpha_i^* < \frac{1}{N}, y_i = \pm 1\right\}$.

An l_2 variant for ν–SVC can also be devised, whose primal is

$$\begin{array}{ll} \min\limits_{\overrightarrow{w},b,\rho,\overrightarrow{\xi}} & \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \nu\rho + \frac{1}{2N}\sum\limits_{i=1}^{N}\xi_i^2 \\ \text{s.t.} & y_i\left(\langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) \geq \rho - \xi_i, \;\; i = 1,\ldots,N,, \end{array} \tag{3.33}$$

where again the constraints $\xi_i \geq 0$ are no longer needed, by a similar reasoning to the one done for l_2–SVC. The Lagrangian for this case is

$$\mathcal{L}\left(\overrightarrow{w}, b, \rho, \overrightarrow{\xi}, \overrightarrow{\alpha}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \nu\rho + \frac{1}{2N}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i\left[y_i(\langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b) - \rho + \xi_i\right]. \tag{3.34}$$

Making its gradient be zero, the only change in (3.29) is

$$\frac{\partial\mathcal{L}}{\partial\xi_i} = \frac{1}{N}\xi_i - \alpha_i = 0 \quad \Rightarrow \quad \alpha_i = \frac{1}{N}\xi_i. \tag{3.35}$$

The dual is finally

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(K_{ij} + N\delta_{ij}\right) \\ \text{s.t.} \quad & \begin{cases} \alpha_i \geq 0, & i = 1, \ldots, N, \\ \sum_{i=1}^{N}\alpha_i y_i = 0, & \\ \sum_{i=1}^{N}\alpha_i = \nu. & \end{cases} \end{aligned} \tag{3.36}$$

Note the analogy with (3.14). The kernel function can be considered again to be slightly modified to include the sum of the squared coefficients. The problem with robustness remains, since the influence of a single point is not bounded, and α_i is proportional to ξ_i by (3.35).

We will see in Chapter 5 that in fact l_1–SVC and ν–SVC are very closely related, and the same happens for l_2–SVC and the l_2 variant of ν–SVC. Next, we explain another model which took its inspiration from l_2–SVC.

3.1.5 *LS*–SVC

Least–Squares Support Vector Classification (LS–SVC) was introduced in [22]. The basic idea is changing the slack variables in (3.10) to

$$\xi_i = 1 - y_i\left(\langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right). \tag{3.37}$$

This allows for negative slack values, so now a square loss is a must if we want to preserve the non–negativity of the primal objective function. Observe also that now the "support hyperplanes" no longer aim to leave each class completely to one side. The aim of the model now is to find two parallel hyperplanes that fit as best as possible the clouds of points of the two classes. Whenever a point does not lie exactly in its support hyperplane it will have a non–zero ξ_i, which will be penalized. Nevertheless, the notion of margin maximization is still kept in the first term of the objective function:

$$\begin{array}{ll} \min\limits_{\overrightarrow{w},b,\overrightarrow{\xi}} & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\sum\limits_{i=1}^{N}\xi_i^2 \\ \text{s.t.} & y_i\left(\langle\overrightarrow{w},\Phi\left(\overrightarrow{x}_i\right)\rangle + b\right) = 1-\xi_i, \quad i = 1\ldots,N. \end{array} \tag{3.38}$$

The choice (3.37) for the slack variables has the immediate consequence that the inequality constraints become now equality constraints. Thus, the Lagrangian coefficients can be negative. Despite the fact that Definition 2.59 used $\overrightarrow{\beta}$ as equality multipliers, here we will still use $\overrightarrow{\alpha}$ for analogy with the rest of formulations. The Lagrangian of (3.38) is

$$\mathcal{L}\left(\overrightarrow{w},b,\overrightarrow{\xi},\overrightarrow{\alpha}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i\left[y_i(\langle\overrightarrow{w},\Phi\left(\overrightarrow{x}_i\right)\rangle + b) - 1 + \xi_i\right]. \tag{3.39}$$

Requiring that its gradient be zero implies

$$\begin{array}{rcl} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} = \overrightarrow{w} - \sum_{i=1}^{N}\alpha_i y_i\Phi\left(\overrightarrow{x}_i\right) = 0 & \Rightarrow & \overrightarrow{w} = \sum_{i=1}^{N}\alpha_i y_i\Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial\mathcal{L}}{\partial b} = -\sum_{i=1}^{N}\alpha_i y_i = 0 & \Rightarrow & \sum_{i=1}^{N}\alpha_i y_i = 0, \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = C\xi_i - \alpha_i = 0 & \Rightarrow & \alpha_i = C\xi_i. \end{array} \tag{3.40}$$

And so the dual becomes

$$\begin{array}{ll} \min\limits_{\overrightarrow{\alpha}} & \frac{1}{2}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(K_{ij} + \frac{\delta_{ij}}{C}\right) - \sum\limits_{i=1}^{N}\alpha_i \\ \text{s.t.} & \sum\limits_{i=1}^{N}\alpha_i y_i = 0, \end{array} \tag{3.41}$$

which is identical to the dual of l_2–SVC (3.14) except for the fact that the coefficients are no longer enforced to be non–negative. The KKT conditions are also simpler, because there are no complementary slackness conditions:

$$\begin{aligned} \overrightarrow{w}^* &= \textstyle\sum_{i=1}^N \alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right), \\ \textstyle\sum_{i=1}^N \alpha_i^* y_i &= 0, \\ y_i\left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &= 1 - \tfrac{\alpha_i^*}{C}, \qquad i = 1, \ldots, N. \end{aligned} \tag{3.42}$$

If we make the same substitutions than in l_2–SVC to project to the augmented feature space, we would obtain a "hard–margin" LS–SVM, whose aim is to find two parallel hyperplanes $\left\langle \overrightarrow{\tilde{w}}^*, \tilde{\Phi}\left(\overrightarrow{x}_i\right)\right\rangle + \tilde{b}^* = \pm 1$, each crossing the points of their corresponding class. It is quite remarkable that such a problem is guaranteed to have a solution, but that is indeed the case, because the points are projected to two parallel lines in that augmented space, as seen in Figure 3.4.

3.2 Support Vector Regression (SVR)

After the conception of SVC, the SVM paradigm was extended to regression tasks. The different variants are described in the same order as for SVC, that is, l_2, l_1, ν and LS–SVR. In the case of l_2 and l_1, an ϵ is added to the name of the machines, for reasons to follow. The hard–margin case is omitted. It will just be explained briefly in the section for l_2–SVR.

3.2.1 l_2–ϵ–SVR

SVR also seeks to find a linear model that approximates a given function, from which our only knowledge is a subsampling of it in the training set. Because it is nearly impossible that the points are collinear, we must allow somehow for errors. Recalling the ϵ–insensitive loss of Equation (1.6), there is a region of width $2\epsilon > 0$ (the so–called ϵ–*tube*) along the decision hyperplane where the approximation is considered good enough and no error is assumed. This seems a sensible choice for SVR, because our hope is that the function is reasonably approximated by a line (in the feature space: remember that this translates to a curve in the input space).

Therefore, slack variables in ϵ–SVR are defined as

$$\xi_i = \max\left\{0, \left|y_i - \left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right)\right| - \epsilon\right\}, \tag{3.43}$$

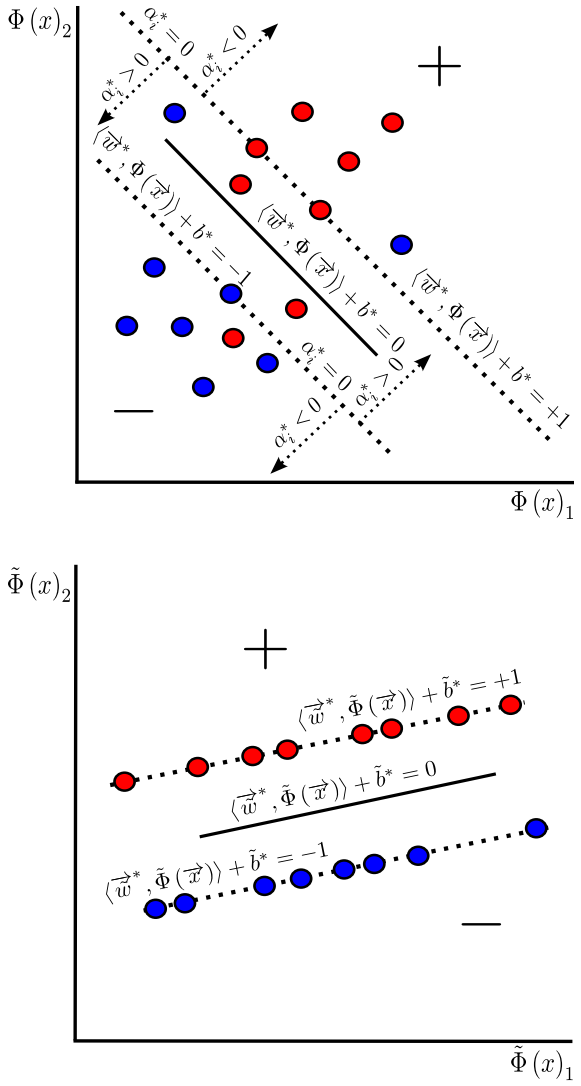

FIGURE 3.4: Depiction of the values of the optimal α_i for *LS*–SVC, both in the feature space and in the augmented feature space. Whereas in the feature space the SVs are the points not lying in the corresponding support hyperplane, in the augmented one all points lie in the support hyperplanes.

l_2–ϵ–SVR penalizes the squares of the slack variables (3.43). This corresponds to using an ϵ–squared loss function in (1.6). The primal problem is formulated as

$$\begin{aligned} \min_{\vec{w},b,\vec{\xi},\vec{\xi}'} \quad & \frac{1}{2}\|\vec{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{N}\left(\xi_i^2 + \left(\xi_i'\right)^2\right) \\ \text{s.t.} \quad & \begin{cases} y_i - \langle \vec{w}, \Phi(\vec{x}_i)\rangle - b \leq \epsilon + \xi_i, & i = 1,\ldots,N, \\ \langle \vec{w}, \Phi(\vec{x}_i)\rangle + b - y_i \leq \epsilon + \xi_i', & i = 1,\ldots,N. \end{cases} \end{aligned} \tag{3.44}$$

Observe that in order to remove the absolute value in (3.43), there are now two different

slack vectors: $\overrightarrow{\xi}$ for quantifying the errors above the tube, and $\overrightarrow{\xi}'$ for the errors below the tube. Note also that, because of the definitions of these vectors, we have the implicit constraints $\xi_i \xi_i' = 0,\ i = 1, \ldots, N$, because a point cannot be at the same time above and below the tube.

The Lagrangian becomes

$$\mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \overrightarrow{\xi}', \overrightarrow{\alpha}, \overrightarrow{\alpha}'\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{N}\left(\xi_i^2 + \left(\xi_i'\right)^2\right) + \sum_{i=1}^{N}\alpha_i\left[y_i - \langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle - b - \epsilon - \xi_i\right] + \sum_{i=1}^{N}\alpha_i'\left[\langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b - y_i - \epsilon - \xi_i'\right].$$

Requiring that its gradient be zero implies

$$\begin{array}{rcl} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} & = & \overrightarrow{w} - \sum_{i=1}^{N}\alpha_i\Phi\left(\overrightarrow{x}_i\right) + \sum_{i=1}^{N}\alpha_i'\Phi\left(\overrightarrow{x}_i\right) = 0 \\ & \Rightarrow & \overrightarrow{w} = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i'\right)\Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial\mathcal{L}}{\partial b} & = & \sum_{i=1}^{N}\left(\alpha_i' - \alpha_i\right) = 0 \ \Rightarrow\ \sum_{i=1}^{N}\left(\alpha_i - \alpha_i'\right) = 0, \\ \frac{\partial\mathcal{L}}{\partial\xi_i} & = & C\xi_i - \alpha_i = 0 \ \Rightarrow\ \alpha_i = C\xi_i, \\ \frac{\partial\mathcal{L}}{\partial\xi_i'} & = & C\xi_i' - \alpha_i' = 0 \ \Rightarrow\ \alpha_i' = C\xi_i', \end{array} \tag{3.45}$$

Combining the two last equations above yields $\alpha_i\alpha_i' = 0$. Substituting (3.45) in the Lagrangian, and using this last fact, produces the dual problem

$$\begin{array}{ll} \min\limits_{\overrightarrow{\alpha}} & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i - \alpha_i'\right)\left(\alpha_j - \alpha_j'\right)\left(K_{ij} + \frac{\delta_{ij}}{C}\right) - \sum_{i=1}^{N}y_i\left(\alpha_i - \alpha_i'\right) + \epsilon\sum_{i=1}^{N}\left(\alpha_i + \alpha_i'\right) \\ \text{s.t.} & \begin{cases} 0 \le \alpha_i, \ \ i = 1, \ldots, N, \\ 0 \le \alpha_i', \ \ i = 1, \ldots, N, \\ \sum_{i=1}^{N}\left(\alpha_i - \alpha_i'\right) = 0, \end{cases} \end{array} \tag{3.46}$$

which is a bit more complicated than (3.14) because of the term with ϵ. Note also that now the targets y_i do not appear in the quadratic term of the objective function, but in the linear one.

The KKT conditions are also a bit more complex than for l_2–SVC:

$$
\begin{aligned}
\overrightarrow{w}^* = \textstyle\sum_{i=1}^{N} \left(\alpha_i^* - (\alpha_i')^*\right) \Phi\left(\overrightarrow{x}_i\right), & \\
\textstyle\sum_{i=1}^{N} \left(\alpha_i^* - (\alpha_i')^*\right) = 0, & \\
\alpha_i^* \geq 0, & \quad i = 1, \ldots, N, \\
(\alpha_i')^* \geq 0, & \quad i = 1, \ldots, N, \\
y_i - \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle - b^* \leq \epsilon + \tfrac{\alpha_i^*}{C}, & \quad i = 1. \ldots, N, \\
\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^* - y_i \leq \epsilon + \tfrac{(\alpha_i')^*}{C}, & \quad i = 1, \ldots, N. \\
\alpha_i^* \left[y_i - \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle - b^* - \epsilon - \tfrac{\alpha_i^*}{C}\right] = 0, & \quad i = 1, \ldots, N, \\
(\alpha_i')^* \left[\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^* - y_i - \epsilon - \tfrac{(\alpha_i')^*}{C}\right] = 0, & \quad i = 1, \ldots, N, \\
\alpha_i^* \left(\alpha_i'\right)^* = 0, & \quad i = 1, \ldots, N.
\end{aligned}
\tag{3.47}
$$

All points with $\alpha_i^* > 0$ or $(\alpha_i')^* > 0$ become SVs and have an influence in the model. Splitting the above in different cases we have:

- $\alpha_i^* = 0$: then $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^* \geq y_i - \epsilon$.
- $(\alpha_i')^* = 0$: then $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^* \leq y_i + \epsilon$.
- $\alpha_i^* > 0$: then $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^* = y_i - \epsilon - \frac{\alpha_i^*}{C} < y_i - \epsilon$.
- $(\alpha_i')^* > 0$: then $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^* = y_i + \epsilon + \frac{\alpha_i^*}{C} > y_i + \epsilon$.

This means that points inside the tube have $\alpha_i^* = (\alpha_i')^* = 0$, whereas points above the tube have $\alpha_i^* > 0$, $(\alpha_i')^* = 0$, and points below the tube have $(\alpha_i')^* > 0$, $\alpha_i^* = 0$. Again, there is a problem with outliers (i.e. points abnormally far from the tube), since their potential influence on the model is unlimited, because the values of α_i are proportional to those of ξ_i.

Similarly to what happened for l_2 and LS–SVC, there is a related hard–margin formulation in the augmented feature space, with no slack variables in (3.44), so that all points are required to be inside the tube or on its borders. The KKT conditions establish for this case that:

- $\alpha_i^* = 0$: then $\left\langle \overrightarrow{\tilde{w}}^*, \tilde{\Phi}\left(\overrightarrow{x}_i\right)\right\rangle + \tilde{b}^* \geq y_i - \epsilon$.
- $(\alpha_i')^* = 0$: then $\left\langle \overrightarrow{\tilde{w}}^*, \tilde{\Phi}\left(\overrightarrow{x}_i\right)\right\rangle + \tilde{b}^* \leq y_i + \epsilon$.
- $\alpha_i^* > 0$: then $\left\langle \overrightarrow{\tilde{w}}^*, \tilde{\Phi}\left(\overrightarrow{x}_i\right)\right\rangle + \tilde{b}^* = y_i - \epsilon$.
- $(\alpha_i')^* > 0$: then $\left\langle \overrightarrow{\tilde{w}}^*, \Phi\left(\overrightarrow{x}_i\right)\right\rangle + \tilde{b}^* = y_i + \epsilon$.

Thus, the SVs are always in the border of the tube, whereas points inside the tube (and possibly some others on the border) are assigned zero coefficients. The l_2 and the hard–margin cases are summarized in Figure 3.5.

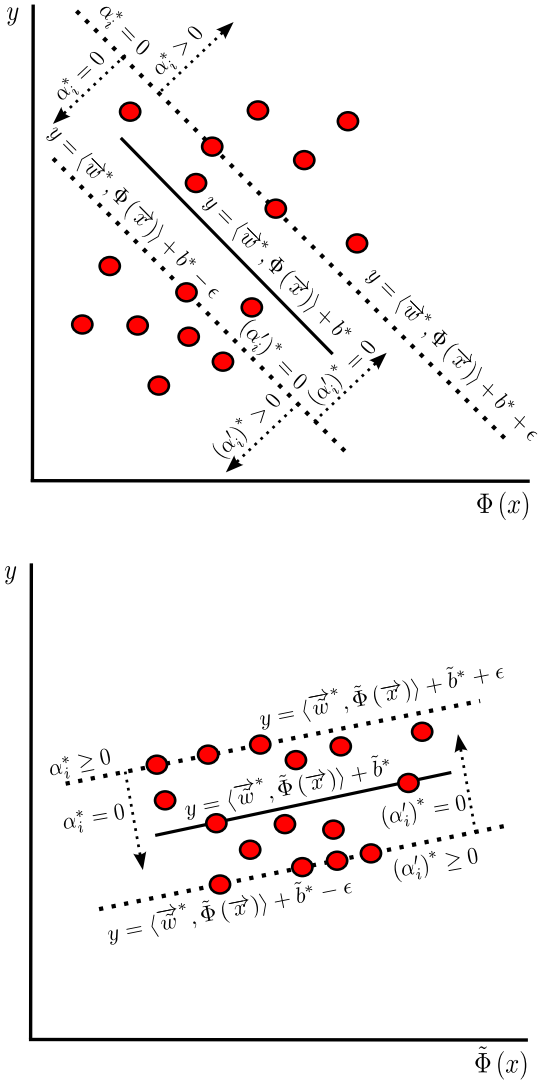

FIGURE 3.5: Depiction of the values of the optimal α_i and α_i' for l_2–SVR, both in the feature space and in the augmented feature space. Whereas in the feature space the SVs are the points outside the tube, in the augmented one they must lie on the borders of the tube.

3.2.2 l_1–ϵ–SVR

Similar in spirit to l_1–SVC, l_1–SVR uses an ϵ–insensitive 1–loss instead of the 2–loss of l_2–SVR), so that the primal problem becomes

$$\begin{aligned} \min_{\overrightarrow{w},b,\overrightarrow{\xi},\overrightarrow{\xi}'} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i'\right) \\ \text{s.t.} \quad & \begin{cases} y_i - \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle - b \leq \epsilon + \xi_i, & i = 1,\ldots,N, \\ \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b - y_i \leq \epsilon + \xi_i', & i = 1,\ldots,N, \\ \xi_i \geq 0, & i = 1,\ldots,N, \\ \xi_i' \geq 0, & i = 1,\ldots,N. \end{cases} \end{aligned} \tag{3.48}$$

As was the case for l_1–SVC, now it is required that the ξ_i and ξ_i' are non–negative, so as to satisfy (3.43). Analogously to what happened in that case, the resulting dual of (3.48) is the same as (3.46), except for an additional upper–bound of C in the Lagrangian coefficients and the removal of the term δ_{ij}/C. In fact, the Lagrangian is

$$\begin{aligned} \mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \overrightarrow{\xi}', \overrightarrow{\alpha}, \overrightarrow{\alpha}', \overrightarrow{\gamma}, \overrightarrow{\gamma}'\right) &= \frac{1}{2}\|\overrightarrow{w}\|_2^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i'\right) \\ &+ \sum_{i=1}^{N}\alpha_i\left[y_i - \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle - b - \epsilon - \xi_i\right] + \sum_{i=1}^{N}\alpha_i'\left[\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b - y_i - \epsilon - \xi_i'\right] \\ &- \sum_{i=1}^{N}\gamma_i\xi_i - \sum_{i=1}^{N}\gamma_i'\xi_i', \end{aligned}$$

and its gradient is zero when (3.45) holds, with the following changes:

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\xi_i} &= C - \alpha_i - \gamma_i = 0 \;\Rightarrow\; 0 \leq \alpha_i \leq C,\; i = 1,\ldots,N, \\ \frac{\partial\mathcal{L}}{\partial\xi_i'} &= C - \alpha_i' - \gamma_i' = 0 \;\Rightarrow\; 0 \leq \alpha_i' \leq C,\; i = 1,\ldots,N. \end{aligned} \tag{3.49}$$

Substituting all this produces the dual

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i - \alpha_i'\right)\left(\alpha_j - \alpha_j'\right)K_{ij} - \sum_{i=1}^{N}y_i\left(\alpha_i - \alpha_i'\right) + \epsilon\sum_{i=1}^{N}\left(\alpha_i + \alpha_i'\right) \\ \text{s.t.} \quad & \begin{cases} 0 \leq \alpha_i \leq C, & i = 1,\ldots,N, \\ 0 \leq \alpha_i' \leq C, & i = 1,\ldots,N, \\ \sum_{i=1}^{N}\left(\alpha_i - \alpha_i'\right) = 0. \end{cases} \end{aligned} \tag{3.50}$$

Observe that, even if $\xi_i \xi_i' = 0,\ i = 1, \ldots, N$ still holds, this no longer implies that $\alpha_i \alpha_i' = 0,\ i = 1, \ldots, N$. However, the KKT conditions show that this is still true:

$$
\begin{aligned}
\overrightarrow{w}^* = \textstyle\sum_{i=1}^N \left(\alpha_i^* - (\alpha_i')^*\right) \Phi\left(\overrightarrow{x}_i\right), & \\
\textstyle\sum_{i=1}^N \left(\alpha_i^* - (\alpha_i')^*\right) = 0, & \\
0 \leq \alpha_i^* \leq C, & \quad i = 1, \ldots, N, \\
0 \leq (\alpha_i')^* \leq C, & \quad i = 1, \ldots, N, \\
y_i - \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle - b^* \leq \epsilon + \xi_i^*, & \quad i = 1 \ldots., N, \\
\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* - y_i \leq \epsilon + (\xi_i')^*, & \quad i = 1, \ldots, N, \\
\xi_i^* \geq 0, & \quad i = 1, \ldots, N, \\
(\xi_i')^* \geq 0, & \quad i = 1, \ldots, N, \\
\alpha_i^* \left[y_i - \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle - b^* - \epsilon - \xi_i^*\right] = 0, & \quad i = 1, \ldots, N, \\
(\alpha_i')^* \left[\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* - y_i - \epsilon - (\xi_i')^*\right] = 0, & \quad i = 1, \ldots, N, \\
\xi_i^* (\xi_i')^* = 0, & \quad i = 1, \ldots, N, \\
(C - \alpha_i^*) \xi_i^* = 0, & \quad i = 1, \ldots, N, \\
\left(C - (\alpha_i')^*\right) (\xi_i')^* = 0, & \quad i = 1, \ldots, N.
\end{aligned}
\tag{3.51}
$$

To see that indeed $\alpha_i \alpha_i^* = 0$, combining the complementary slackness conditions with α_i and α_i^* shows that, if we had $\alpha_i^* > 0$ and $(\alpha_i')^* > 0$, we would have $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* = y_i + \epsilon + (\xi_i')^* = y_i - \epsilon - \xi_i^*$, so that $\xi_i^* + (\xi_i')^* = -2\epsilon$. If $\epsilon > 0$ this of course is impossible, so that $\alpha_i^* (\alpha_i')^* = 0$ and we are done. In the extreme case with $\epsilon = 0$, it is possible, provided that we take $\xi_i^* = (\xi_i')^* = 0$. Assume that $\alpha_i^* > 0$ and $(\alpha_i')^* > 0$. Then, we can subtract $\delta = \min\left\{\alpha_i^*, (\alpha_i')^*\right\}$ from both coefficients and still have a feasible solution, because the dual constraints in (3.50) still hold. To see that this solution is still optimal, it suffices to realize that $(\alpha_i^* - \delta) - \left((\alpha_i')^* - \delta\right) = \alpha_i^* - (\alpha_i')^*$ and the dual function remains the same, because the only term which does not depend on this difference is the one with ϵ, which is always 0, and again we are done.

Thus, there are 6 possible different cases, which are summarized in Figure 3.6:

- $\alpha_i^* = 0$: then $\xi_i^* = 0$ and $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* \geq y_i - \epsilon$.
- $(\alpha_i')^* = 0$: then $(\xi_i')^* = 0$ and $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* \leq y_i + \epsilon$.
- $0 < \alpha_i^* < C$: then $\xi_i^* = 0$ and $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* = y_i - \epsilon$.
- $0 < (\alpha_i')^* < C$: then $(\xi_i')^* = 0$ and $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* = y_i + \epsilon$.
- $\alpha_i^* = C$: then $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* \leq y_i - \epsilon$.
- $(\alpha_i')^* = C$: then $\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + b^* \geq y_i + \epsilon$.

Observe that the SVs are now the points that are placed on the borders of the tube or outside from it. In the first case, their coefficients can have whichever value (there may also be some points on the borders which are not SVs), whereas if they are outside the tube their coefficient is C. Once more, these points are SVs because they are informative points: it is expected that most points will lie inside the tube, thereby having a zero coefficient.

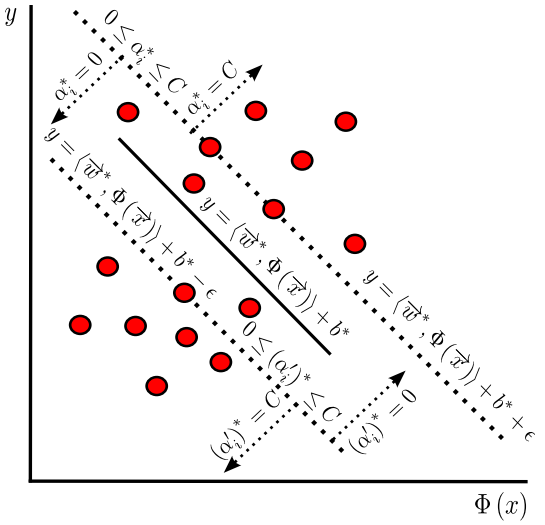

FIGURE 3.6: Depiction of the values of the optimal α_i and α_i' for l_1–SVR. The SVs are the points which are not strictly inside the tube.

The optimal bias b^* is obtained from the points on the borders. Averaging the result gives

$$b^* = \frac{1}{2}\left(\frac{1}{|S|}\sum_{S}\left(y_i - \epsilon - \langle \vec{w}^*, \Phi(\vec{x}_i)\rangle\right) + \frac{1}{|S'|}\sum_{S'}\left(y_i + \epsilon - \langle \vec{w}^*, \Phi(\vec{x}_i)\rangle\right)\right), \quad (3.52)$$

where S is the set of indices $\{i : 0 < \alpha_i^* < C\}$ and $S' = \{i : 0 < (\alpha_i')^* < C\}$.

3.2.3 ν–SVR

If we recall the justification of ν–SVC, the main idea was to remove the parameter C, which is difficult to tune, and use instead an alternative parameter $\nu \in [0, 1]$. An l_1–ϵ–SVR is even more difficult to tune, as it makes use of another parameter ϵ, whose optimal value is not known a priori. Broadly speaking, the larger ϵ the wider the tube, thus the sparser the model and the less sensitive to errors. Conversely, in the limit case of $\epsilon = 0$ every point not lying exactly on the decision boundary incurs in an error.

In ν–SVR, the regularization parameter C is kept, and the introduction of ν is aimed so as not to tune ϵ. In fact, ϵ becomes now a primal variable, and it is therefore optimized automatically. In such a way, we can establish a priori a value for C, and then change ν to trade off the size of the tube against model complexity:

$$\begin{aligned} \min_{\overrightarrow{w},b,\epsilon,\xi,\xi'} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + C\left(\nu\epsilon + \frac{1}{N}\sum_{i=1}^{N}\left(\xi_i + \xi_i'\right)\right) \\ \text{s.t.} \quad & \begin{cases} y_i - \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle - b \leq \epsilon + \xi_i, & i = 1\ldots, N, \\ \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b - y_i \leq \epsilon + \xi_i', & i = 1, \ldots, N, \\ \xi_i \geq 0, & i = 1, \ldots, N, \\ \xi_i' \geq 0, & i = 1, \ldots, N, \\ \epsilon \geq 0. \end{cases} \end{aligned} \tag{3.53}$$

The Lagrangian of (3.53) is

$$\begin{aligned} \mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \overrightarrow{\xi}', \epsilon, \overrightarrow{\alpha}, \overrightarrow{\alpha}', \overrightarrow{\gamma}, \overrightarrow{\gamma}', \eta\right) &= \frac{1}{2}\|\overrightarrow{w}\|_2^2 + C\left(\nu\epsilon + \frac{1}{N}\sum_{i=1}^{N}\left(\xi_i + \xi_i'\right)\right) \\ &+ \sum_{i=1}^{N}\alpha_i\left[y_i - \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle - b - \epsilon - \xi_i\right] + \sum_{i=1}^{N}\alpha_i'\left[\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b - y_i - \epsilon - \xi_i'\right] \\ &- \sum_{i=1}^{N}\gamma_i\xi_i - \sum_{i=1}^{N}\gamma_i'\xi_i' - \eta\epsilon, \end{aligned}$$

Equaling the gradient to zero produces

$$\begin{array}{rcl} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} & = & \overrightarrow{w} - \sum_{i=1}^{N}\alpha_i\Phi\left(\overrightarrow{x}_i\right) + \sum_{i=1}^{N}\alpha_i'\Phi\left(\overrightarrow{x}_i\right) = 0 \\ & \Rightarrow & \overrightarrow{w} = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i'\right)\Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial\mathcal{L}}{\partial b} & = & \sum_{i=1}^{N}\left(\alpha_i' - \alpha_i\right) = 0 \;\Rightarrow\; \sum_{i=1}^{N}\left(\alpha_i - \alpha_i'\right) = 0, \\ \frac{\partial\mathcal{L}}{\partial\xi_i} & = & \frac{C}{N} - \alpha_i - \gamma_i = 0 \;\Rightarrow\; 0 \leq \alpha_i \leq \frac{C}{N}, \\ \frac{\partial\mathcal{L}}{\partial\xi_i'} & = & \frac{C}{N} - \alpha_i' - \gamma_i' = 0 \;\Rightarrow\; 0 \leq \alpha_i' \leq \frac{C}{N}, \\ \frac{\partial\mathcal{L}}{\partial\epsilon} & = & C\nu - \sum_{i=1}^{N}\left(\alpha_i + \alpha_i'\right) - \eta = 0 \;\Rightarrow\; \sum_{i=1}^{N}\left(\alpha_i + \alpha_i'\right) \leq C\nu, \end{array} \tag{3.54}$$

giving the dual

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i-\alpha_i'\right)\left(\alpha_j-\alpha_j'\right)K_{ij}-\sum_{i=1}^{N}y_i\left(\alpha_i-\alpha_i'\right) \\ \text{s.t.} \quad & \begin{cases} 0\leq\alpha_i\leq\frac{C}{N}, & i=1,\ldots,N, \\ 0\leq\alpha_i'\leq\frac{C}{N}, & i=1,\ldots,N, \\ \sum_{i=1}^{N}\left(\alpha_i-\alpha_i'\right)=0, & \\ \sum_{i=1}^{N}\left(\alpha_i+\alpha_i'\right)\leq C\nu. & \end{cases} \end{aligned} \tag{3.55}$$

The last inequality constraint can be assumed to be an equality one, because of the results in [48]. In the primal this means that the constraint $\epsilon \geq 0$ is redundant, as was the constraint $\rho \geq 0$ in ν–SVC. This fact will become important in Chapter 6 to include this problem under a common formulation. In any case, the KKT conditions turn out to be

$$\begin{aligned} \overrightarrow{w}^* &= \textstyle\sum_{i=1}^{N}\left(\alpha_i^*-\left(\alpha_i'\right)^*\right)\Phi\left(\overrightarrow{x}_i\right), & \\ \textstyle\sum_{i=1}^{N}\left(\alpha_i^*-\left(\alpha_i'\right)^*\right) &= 0, & \\ \textstyle\sum_{i=1}^{N}\left(\alpha_i^*+\left(\alpha_i'\right)^*\right) &\leq C\nu, & \\ 0\leq\alpha_i^* &\leq\tfrac{C}{N}, & i=1,\ldots,N, \\ 0\leq\left(\alpha_i'\right)^* &\leq\tfrac{C}{N}, & i=1,\ldots,N, \\ y_i-\langle\overrightarrow{w}^*,\Phi\left(\overrightarrow{x}_i\right)\rangle-b^* &\leq\epsilon+\xi_i^*, & i=1\ldots,N, \\ \langle\overrightarrow{w}^*,\Phi\left(\overrightarrow{x}_i\right)\rangle+b^*-y_i &\leq\epsilon+\left(\xi_i'\right)^*, & i=1,\ldots,N, \\ \xi_i^* &\geq 0, & i=1,\ldots,N, \\ \left(\xi_i'\right)^* &\geq 0, & i=1,\ldots,N, \\ \alpha_i^*\left[y_i-\langle\overrightarrow{w}^*,\Phi\left(\overrightarrow{x}_i\right)\rangle-b^*-\epsilon-\xi_i^*\right] &= 0, & i=1,\ldots,N, \\ \left(\alpha_i'\right)^*\left[\langle\overrightarrow{w}^*,\Phi\left(\overrightarrow{x}_i\right)\rangle+b^*-y_i-\epsilon-\left(\xi_i'\right)^*\right] &= 0, & i=1,\ldots,N, \\ \xi_i^*\left(\xi_i'\right)^* &= 0, & i=1,\ldots,N, \\ \left(\tfrac{C}{N}-\alpha_i^*\right)\xi_i^* &= 0, & i=1,\ldots,N, \\ \left(\tfrac{C}{N}-\left(\alpha_i'\right)^*\right)\left(\xi_i'\right)^* &= 0, & i=1,\ldots,N. \end{aligned} \tag{3.56}$$

Similarly to the case for classification, if we take the ϵ–insensitive squared loss in (3.53), the dual is identical to (3.55) except for two single aspects:

- The quadratic term is changed to $\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i-\alpha_i')(\alpha_j-\alpha_j')\left(K_{ij}+\frac{N}{C}\delta_{ij}\right)$.
- The Lagrangian coefficients α_i and α_i' are no longer upper bounded by $\frac{C}{N}$.

3.2.4 *LS*–SVR

Least–Squares Support Vector Regression (LS–SVR) is very similar to LS–SVC. The slacks are now the difference between the target value and the output of the regressor:

$$\xi_i = y_i - \left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b\right). \tag{3.57}$$

This allows again for negative slack values, so it is not necessary to add variables ξ_i' or to distinguish between errors below and above the tube. In fact, there is not such a tube, because the ϵ parameter is removed. Whenever a point does not lie exactly in the regression line it will have a non–zero ξ_i, which will be penalized. This corresponds to taking a pure squared loss ($\epsilon = 0$) in Figure 1.2:

$$\begin{array}{ll} \min\limits_{\overrightarrow{w},b,\overrightarrow{\xi}} & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\sum\limits_{i=1}^{N}\xi_i^2 \\ \text{s.t.} & \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b = y_i - \xi_i, \quad i = 1\ldots, N. \end{array} \tag{3.58}$$

This primal problem has the following Lagrangian:

$$\mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \overrightarrow{\alpha}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i\left[\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b - y_i + \xi_i\right]. \tag{3.59}$$

If we enforce its gradient to become 0:

$$\begin{array}{rcl} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} = \overrightarrow{w} - \sum_{i=1}^{N}\alpha_i\Phi\left(\overrightarrow{x}_i\right) = 0 & \Rightarrow & \overrightarrow{w} = \sum_{i=1}^{N}\alpha_i\Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial\mathcal{L}}{\partial b} = -\sum_{i=1}^{N}\alpha_i = 0 & \Rightarrow & \sum_{i=1}^{N}\alpha_i = 0, \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = C\xi_i - \alpha_i = 0 & \Rightarrow & \alpha_i = C\xi_i. \end{array} \tag{3.60}$$

And so the dual becomes

$$\begin{array}{ll} \min\limits_{\overrightarrow{\alpha}} & \frac{1}{2}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\alpha_i\alpha_j\left(K_{ij} + \frac{\delta_{ij}}{C}\right) - \sum\limits_{i=1}^{N}\alpha_i y_i \\ \text{s.t.} & \sum\limits_{i=1}^{N}\alpha_i = 0, \end{array} \tag{3.61}$$

whereas the KKT conditions are

$$\begin{aligned} \overrightarrow{w}^* &= \textstyle\sum_{i=1}^N \alpha_i^* \Phi\left(\overrightarrow{x}_i\right), \\ \textstyle\sum_{i=1}^N \alpha_i^* &= 0, \\ \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^* &= y_i - \tfrac{\alpha_i^*}{C}, \qquad i = 1, \ldots, N. \end{aligned} \tag{3.62}$$

3.3 Unsupervised Learning: One–Class Support Vector Machines (OC)

One–Class Support Vector Machines [49] will constitute the only example of unsupervised learning considered in this work. Recall from Chapter 1 that in unsupervised learning the data are unlabeled and hence our goal is to learn the "trend" that these data follow. Several points of view are possible here. If we suspect that the data belong to different "classes" or "groups" and we seek to distinguish between these groups, we are adopting a *clustering* viewpoint. If we want to infer the probability distribution generating the data, then we are taking into consideration *density estimation*.

OC is more related to density estimation than to clustering. The idea is delimiting a region of the input space where the probability density is estimated to be high. It is hoped that all the points (or at least most of them) will lie in that area. The relationship with SVMs stems from the fact that the limit of this region is in fact a hyperplane in the feature space (which translates to a hypersurface in the input space). From this hyperplane we can build a somewhat artificial classifier, whose convention is giving $+1$ for points inside this region and -1 for those outside.

In order to find this hyperplane, the primal formulation to be solved is

$$\begin{aligned} \min_{\overrightarrow{w},\rho,\overrightarrow{\xi}} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \rho + \frac{1}{\nu N}\sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & \begin{cases} \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle \geq \rho - \xi_i, & i = 1, \ldots, N, \\ \xi_i \geq 0, & i = 1, \ldots, N, \end{cases} \end{aligned} \tag{3.63}$$

Thus, the reference hyperplane is $\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle - \rho = 0$, so ρ can be regarded as a kind of bias. Whenever a point is not in the correct side of this hyperplane, that is, when $\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle < \rho$, the error is quantified by the corresponding slack ξ_i. The parameter ν acts again as trade–off between the margin of the hyperplane and the number and magnitude of errors.

Taking the Lagrangian of (3.63), which turns out to be

$$\mathcal{L}\left(\overrightarrow{w}, b, \rho, \overrightarrow{\xi}, \overrightarrow{\alpha}, \overrightarrow{\gamma}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[\langle\overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle - \rho + \xi_i\right] - \sum_{i=1}^{N}\gamma_i\xi_i, \tag{3.64}$$

and equaling its gradient to zero produces

$$\begin{array}{rcl} \frac{\partial\mathcal{L}}{\partial\overrightarrow{w}} = \overrightarrow{w} - \sum_{i=1}^{N}\alpha_i\Phi\left(\overrightarrow{x}_i\right) = 0 & \Rightarrow & \overrightarrow{w} = \sum_{i=1}^{N}\alpha_i\Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial\mathcal{L}}{\partial\rho} = -1 + \sum_{i=1}^{N}\alpha_i = 0 & \Rightarrow & \sum_{i=1}^{N}\alpha_i = 1, \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = \frac{1}{\nu N} - \alpha_i - \gamma_i = 0 & \Rightarrow & 0 \leq \alpha_i \leq \frac{1}{\nu N}. \end{array} \tag{3.65}$$

Substitution yields the dual

$$\begin{array}{ll} \min\limits_{\overrightarrow{\alpha}} & \frac{1}{2}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\alpha_i\alpha_j K_{ij} \\ \text{s.t.} & \begin{cases} 0 \leq \alpha_i \leq \frac{1}{\nu N}, & i = 1, \ldots, N, \\ \sum_{i=1}^{N}\alpha_i = 1, \end{cases} \end{array} \tag{3.66}$$

which is quite similar to the one for l_1–ν–SVC. So are the KKT conditions:

$$\begin{array}{rll} \overrightarrow{w}^* = \sum_{i=1}^{N}\alpha_i^*\Phi\left(\overrightarrow{x}_i\right), & & \\ \sum_{i=1}^{N}\alpha_i^* = 1, & & \\ 0 \leq \alpha_i^* \leq \frac{1}{\nu N}, & i = 1, \ldots, N, & \\ \langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle \geq \rho^* - \xi_i^*, & i = 1, \ldots, N, & \\ \xi_i^* \geq 0, & i = 1, \ldots, N, & \\ \alpha_i^*\left[\langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle - \rho^* + \xi_i^*\right] = 0, & i = 1, \ldots, N, & \\ \left(\frac{1}{\nu N} - \alpha_i^*\right)\xi_i^* = 0, & i = 1, \ldots, N. & \end{array} \tag{3.67}$$

Separating in cases we get

- $\alpha_i^* = 0$: then $\xi_i^* = 0$, so $\langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle \geq \rho^*$.
- $0 < \alpha_i^* < \frac{1}{\nu N}$: then $\xi_i^* = 0$ and $\langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle = \rho^*$.
- $\alpha_i^* = \frac{1}{\nu N}$: then $\langle\overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle = \rho^* - \xi_i^* \leq \rho^*$,

which are illustrated in Figure 3.7. Once more, the SVs are the atypical or most informative points, in this case the ones lying either on the hyperplane or in the "wrong" side of it. The optimal ρ^* is calculated from the points with $0 < \alpha_i^* < \frac{1}{\nu N}$.

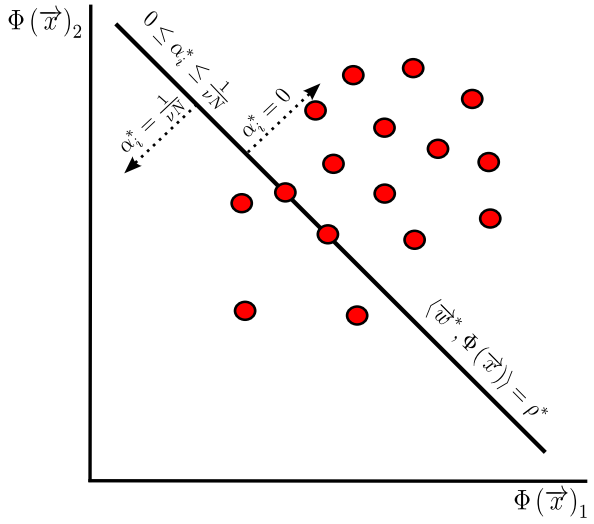

FIGURE 3.7: Depiction of the values of the optimal α_i for OC. The SVs are points which are not in the "most likely" side of the hyperplane.

Taking again a square loss in the primal (3.63) results in the same dual as (3.66), except for the facts that:

- The objective function is changed to $\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\left(K_{ij}+\nu N\delta_{ij}\right)$.
- The Lagrangian coefficients α_i are no longer upper bounded by $\frac{1}{\nu N}$.

3.4 Related Geometric Formulations

Some geometric problems can also be extended to the feature space, as the only operations involved among points are calculations of distances, angles and projections, which can be written in terms of norms and inner products. In this thesis we will center on the so–called Minimum Norm and Nearest Point problems. These problems have an important domain of application in Robotics, in the discipline termed *collision detection* [50].

It will turn out that these geometric problems have a very close relationship to the SVM formulations described in the previous section. However, we will postpone the study of this relationship to Chapter 5. In this section we just formulate and explain the problems in the same fashion than the SVC, SVR and OC ones.

3.4.1 Minimum Norm Problem (MNP)

We begin with a simple problem, the so–called Minimum Norm Problem. It consists of finding the point of a polyhedron closest to the origin of coordinates. In this context, by "polyhedron" we mean the convex hull of the patterns of the sample. Recalling Definition 2.42 and using a similar notation than in SVMs, a point in this convex hull is of the form $\overrightarrow{w} = \sum_{i=1}^{N} \alpha_i \overrightarrow{x}_i$, with $\alpha_i \geq 0 \ \forall i$ and $\sum_{i=1}^{N} \alpha_i = 1$. If we are working in the feature space, there is no change but for the fact that now $\overrightarrow{w} = \sum_{i=1}^{N} \alpha_i \Phi(\overrightarrow{x}_i)$.

From now on, we will assume that we are working in this feature space. Using the l_2–norm as the measure to know which is the closest point, it is straightforward that the problem can then be formulated as

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K_{ij} \\ \text{s.t.} \quad & \begin{cases} 0 \leq \alpha_i, & i = 1, \ldots, N, \\ \sum_{i=1}^{N} \alpha_i = 1, \end{cases} \end{aligned} \tag{3.68}$$

due to the fact that the objective function is equivalent to $\frac{1}{2} \|\overrightarrow{w}\|_2^2$.

A slight variation for this problem is obtained when we consider the reduced convex hull of the patterns of the sample instead of the convex hull itself. Taking a look at Definition 2.45, this only influences in the upper–bound μ for the coefficients, so the problem becomes

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K_{ij} \\ \text{s.t.} \quad & \begin{cases} 0 \leq \alpha_i \leq \mu, & i = 1, \ldots, N, \\ \sum_{i=1}^{N} \alpha_i = 1. \end{cases} \end{aligned} \tag{3.69}$$

We will refer to problem (3.68) as CH–MNP (Convex Hull Minimum Norm Problem), whereas (3.69) will be referred to as RCH–MNP. A depiction of the optimal solution for a simple example with 5 patterns is shown in Figure 3.8. In this example, the extreme points of the standard convex hull are the patterns themselves. For RCH–MNP the value of μ is $1/2$, so the extreme points of the reduced convex hull are the midpoints of the segments joining the patterns.

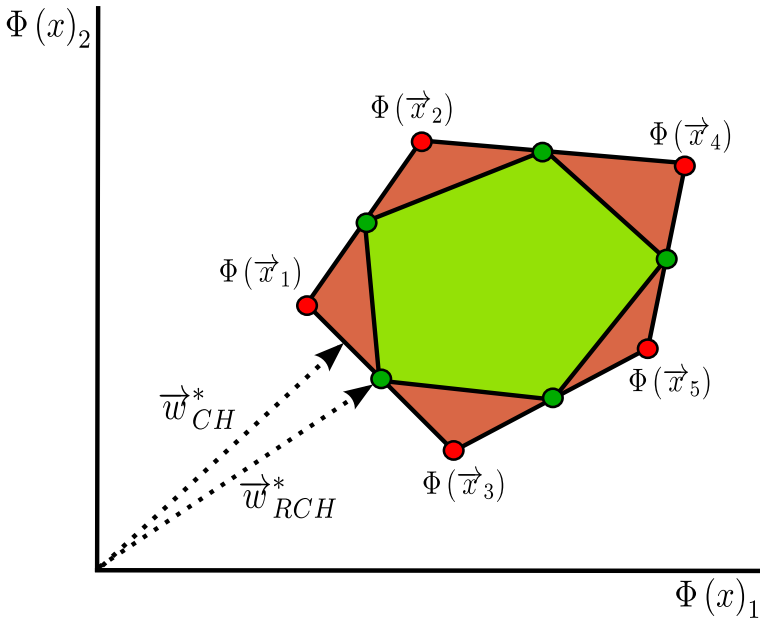

FIGURE 3.8: Illustration of the optimal solutions of CH–MNP and RCH–MNP for a simple example. The convex hull is colored in red and the reduced convex hull in green.

3.4.2 Nearest Point Problem (NPP)

This problem is a generalization of the MNP one. In MNP we did not care about the labels of the patterns, but just built the (reduced) convex hull associated to them. Here, we will distinguish between the positive and the negative instances and build two (reduced) convex hulls, one for each class. Our goal is to find the two closest points in these hulls, hence its name Nearest Point Problem.

To this aim, let us assume, without loss of generality, that any feasible solution $\overrightarrow{w}$ will point towards the positive class. Then, we can write this solution as the difference of two vectors: $\overrightarrow{w} = \overrightarrow{w}_+ - \overrightarrow{w}_-$, such that $\overrightarrow{w}_+$ lies in the positive hull and $\overrightarrow{w}_-$ does so in the negative hull.

Therefore, we can write

$$\overrightarrow{w} = \sum_{\mathcal{I}_+} \alpha_i \Phi(\overrightarrow{x}_i) - \sum_{\mathcal{I}_-} \alpha_i \Phi(\overrightarrow{x}_i),$$

where $\mathcal{I}_\pm$ denotes the indices of the patterns belonging to the positive and negative classes, respectively. In order for $\overrightarrow{w}_+$ and $\overrightarrow{w}_-$ to lie in the hulls, it is required that $\alpha_i \geq 0 \; \forall i$ and $\sum_{\mathcal{I}_+} \alpha_i = \sum_{\mathcal{I}_-} \alpha_i = 1$. If we are working with reduced convex hulls, there is the additional requirement that $\alpha_i \leq \mu \; \forall i$.

Since $y_i = +1$ for patterns in the positive class and $y_i = -1$ for patterns in the negative class, we can write $\overrightarrow{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\overrightarrow{x}_i)$, so that the problem can be formulated for standard convex hulls as

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K_{ij} \\ \text{s.t.} \quad & \begin{cases} 0 \leq \alpha_i, & i = 1, \ldots, N, \\ \sum_{\mathcal{I}_+} \alpha_i = \sum_{\mathcal{I}_-} \alpha_i = 1. \end{cases} \end{aligned} \tag{3.70}$$

Problem (3.70) will be referred to as CH–NPP, in contrast to RCH–NPP, which is the version with reduced convex hulls, which becomes

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K_{ij} \\ \text{s.t.} \quad & \begin{cases} 0 \leq \alpha_i \leq \mu, & i = 1, \ldots, N, \\ \sum_{\mathcal{I}_+} \alpha_i = \sum_{\mathcal{I}_-} \alpha_i = 1. \end{cases} \end{aligned} \tag{3.71}$$

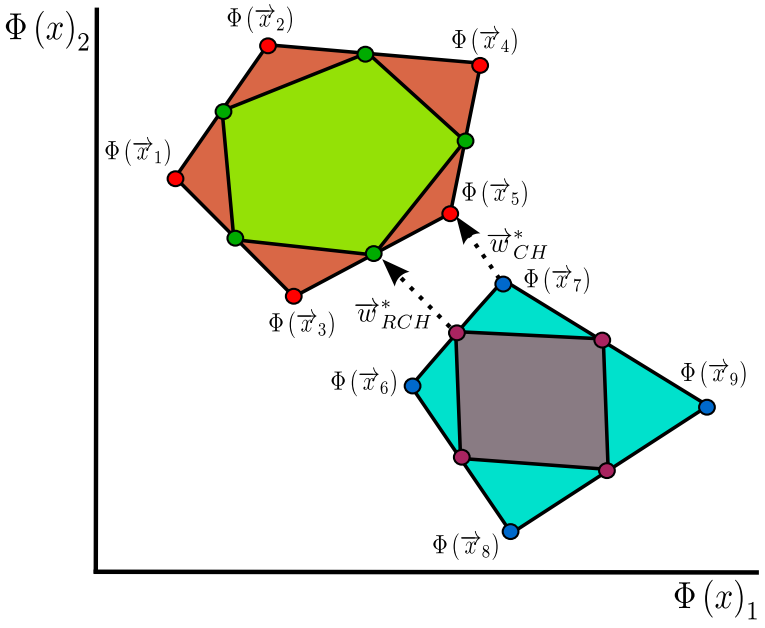

FIGURE 3.9: Illustration of the optimal solutions of CH–NPP and RCH–NPP for a simple example. The positive convex hull is colored in red and the negative one in blue. The reduced convex hulls ($\mu = \frac{1}{2}$) are colored in green and purple, respectively.

Clearly, NPP reduces to MNP whenever one of the classes has as its single point the origin of coordinates. Because the solution does not change when a translation is performed, the same happens when a class has only one point, regardless whether it is the origin of coordinates or not.

A depiction of the optimal solution for a simple example of CH–NPP and RCH–NPP is shown in Figure 3.9. This example has 5 patterns in the positive class and 4 in the negative one. Notice that the solutions for each case need not be parallel, because the reduced convex hulls can have a quite different shape from the one of standard convex hulls.

Chapter 4

State–of–the–art Algorithms

In Chapter 3 we described different SVM formulations, together with some related geometric formulations. It was said that there is a strong relationship between these two kinds of formulations. We did not delve into the details of this relationship, though. In the present chapter we will keep on considering them separately, describing some state–of–the–art algorithms which will be the central topic of the rest of this dissertation, namely the SMO algorithm for SVM training and the MDM algorithm for nearest point problems. For the latter problems, we will also consider a classical algorithm called GSK.

The algorithms are described by means of a common viewpoint and notation. This unification is very convenient, because the original algorithms were described in the different references with different notations and points of view. The present chapter and its common viewpoint pave the way for Chapter 5, where the relationship between SVM and geometric formulations will not only be clarified, but also generalized to include all the formulations of Chapter 3 into a general problem.

4.1 SVM Algorithms

Let us begin with the algorithms used for SVM training. Recalling the different formulations for SVC, SVR and OC introduced in Chapter 3, it is readily seen that their primal formulations are convex programs (cf. Definition 2.58), because their objective functions are convex, their inequality constraints are also convex, and their equality constraints are affine.

This is one of the main advantages of SVMs compared to other Machine Learning paradigms that amount to solve optimization problems, such as Neural Networks. Convexity guarantees the inexistence of local optima: if the primal objective function has a minimum, it must be a global minimum. This is not so in Neural Networks: one of the dangers while training them is getting stuck in local minima. Moreover, reaching the global minimum in SVMs guarantees good generalization properties, because of the SRM principle outlined in Chapter 1.

However, we must keep in mind that the primal formulations are only solvable in the linear case (that is, with a kernel that is the standard inner product), because usually we will not have any clue about what the feature map $\Phi(\cdot)$ is. To make things worse, in some cases this feature map is (potentially) infinite-dimensional, as happens with the Gaussian kernel of Definition 2.37 [2]. Solving the primal implies finding the optimal weight vector $\overrightarrow{w}^*$, which in such a case can also be infinite-dimensional, hence impossible to store computationally.

Fortunately, the dual formulations are a way out of this. They are simpler than the primal formulations, because they are not only convex, but also quadratic programs (cf. Definition 2.57). Looking back at Chapter 3, it can be seen that in fact all the dual objective functions are quadratic and all the dual constraints are affine. Besides, the problem with the unknown feature maps disappears. In the duals we only make use of a kernel function, with the property that $k(\overrightarrow{x}_i, \overrightarrow{x}_j) = \langle \Phi(\overrightarrow{x}_i), \Phi(\overrightarrow{x}_j) \rangle$, as was broadly described in Chapter 2.

Therefore, in practically all the cases it is the dual formulations that are solved, not the primal ones. The only exception are linear SVMs, which will be treated briefly in §4.1.2. For non–linear ones, that is, SVMs with a kernel different from the standard inner product, the approach that has been given most attention are *decomposition algorithms*, which are treated in §4.1.1. The best–known algorithm of this kind is *Sequential Minimal Optimization* (henceforth SMO), which is treated in §4.1.1.1. Some other decomposition algorithms are explained in §4.1.1.2. Other algorithms not based on the idea of decomposition are cited in §4.1.2.

4.1.1 SMO and Decomposition Algorithms

Once we have committed ourselves to solve the dual formulations, this looks like a reasonably easy task. The dual objective functions are quadratic, so they are twice continuously differentiable. This brings to mind several classical algorithms based on gradient computation, which are briefly outlined next [38, 40]:

- *Gradient descent* is the simplest method. Given the negative gradient, which is the steepest direction for minimizing, we advance as much as possible in that direction. Then we calculate the gradient again, and repeat the same procedure until a convergence criterion is met. Convergence is guaranteed, but usually it is a slow one, because the algorithm tends to zig–zag on its way towards the minimum.

- *Conjugate gradient* is especially intended for quadratic functions with a positive definite matrix A. The idea is to optimize over directions $\overrightarrow{v}_i$ that are *conjugate* with respect to A, that is, $\langle \overrightarrow{v}_i, A\overrightarrow{v}_{i+1} \rangle = 0$, $i = 1, \ldots, N$. If the advance factors are carefully chosen, the method can converge in just N iterations.

- *Quasi–Newton* methods try to use both gradient (*first order*) and Hessian (*second order*) information to choose better directions than pure gradient descent. Since the Hessian is expensive to calculate, an approximation to it is estimated. Again, if the advance factors and Hessian updates are carefully chosen, convergence can be guaranteed, as long as the function has a quadratic Taylor expansion around its optimum.

The methods above have been enhanced over the years and successfully used for optimization of convex functions. For instance, among conjugate gradient methods the best–known variants are the Fletcher–Reeves [38] and the Polak–Ribière [51] ones. Among Quasi–Newton ones we have the Davidon–Fletcher–Powell (DFP) [52] and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) [53] variants.

However, the first difficulty comes from the fact that our problems are constrained and the above methods are meant for unconstrained optimization. Ideally, we would like to use some kind of method especially tailored to quadratic programming (QP) or convex programming, because all dual formulations lie in this category. We could think then that some widely available packages for this aim, such as CVX [54] or MOSEK [55], suffice.

This is rarely the case, though. Some other difficulties arise, such as the fact that we are minimizing on a vector $\overrightarrow{\alpha} \in \mathbb{R}^N$. This means that the quadratic term of the dual objective function includes a kernel matrix K which is $N \times N$. These general packages require that the whole matrix be stored in memory. When we are dealing with a dataset with a large number of patterns (what is termed as *large–scale learning*), this soon becomes infeasible. For example, a dataset with 10^5 patterns (which is considered to be medium–sized nowadays), assuming that we need 8 bytes to keep a real number, would require a memory of $10^5 \times 10^5 \times 2^3 = 10^{10} \times 2^3 \approx 2^{30} \times 10 \times 2^3 = 80 \times 2^{30}$ bytes, that is, 80 GB, which is quite a requirement.

As massive flows of data have arisen in applications such as banking and web mining, it is quite common these days to find datasets with millions of patterns. Obviously, for such datasets the huge memory requirements preclude the use of general packages. In addition, these packages do not use any specific information about the problems at hand. In fact, the SVM duals are quite simple, because the constraints are normally reduced to box constraints of the form $l \leq \alpha_i \leq u$, with l and u the lower and upper bounds, and simple linear constraints on $\overrightarrow{\alpha}$.

Another matter of concern is the computational cost. Assuming that we have somehow solved the memory requirements, we would also like some kind of method that solves the dual problem, or at least gives a "reasonable" approximation to the optimum, in a "reasonable" amount of time. We will later deal with what "reasonable" means for an approximation to the optimum. In terms of speed, it is desirable not to surpass a complexity of $\mathcal{O}\left(N^2\right)$. A cost of, for example, $\mathcal{O}\left(N^3\right)$ means that, even for medium–scale learning we would need quite a lot of time to solve some problems.

This additional requirement on complexity restricts much what we can do: for example, it precludes in principle the use of any method that uses second order information, because computing the inverse of the Hessian cannot be done in less than $\mathcal{O}\left(N^3\right)$. In short, what we are looking for is an algorithm that satisfies the following:

- Modest memory requirements: a storage of $O\left(N^2\right)$ is unacceptable for large–scale learning.
- Fast enough: a cost of $\mathcal{O}\left(N^3\right)$ is unacceptable for large and medium–scale learning.
- Especially tailored to the dual formulations at hand: this is expected to simplify the method, despite losing generality.
- Guaranteed to converge: it should converge to an optimum, ideally in a finite number of iterations. Otherwise at least asymptotically, so that we can stop and be close enough to an optimum.

In the rest of this section we will assume that we want to solve the dual of l_1–SVC, that is, problem (3.24). We do this for the sake of completeness (in this case the upper bound u is a finite value of C, and not $+\infty$ as in the l_2 versions) and simplicity at the same time (the SVR formulations are somewhat more involved). As already mentioned, it will not be until Chapter 5 that we will deal with a general problem that subsumes the plethora of formulations of Chapter 3.

With these objectives in mind, the idea of *decomposition* turned out to be a milestone in SVM training. This idea was first outlined by Vapnik et al. in [56] in their *chunking*

algorithm, where it was observed that if we were somehow able to remove from the beginning the points which are not support vectors in the optimum (i.e. the points for which $\alpha_i^* = 0$), then the solution obtained would be the same as the one for the full problem.

Having observed this, the basic concept behind chunking is this motto: "let us just optimize on some coefficients in every iteration, keeping the rest fixed". The subset of coefficients chosen for optimization is called *working set*. Denoting it by B, and referring to the rest of coefficients as $\overline{B} = \{1, \ldots, N\} \setminus B$, in every iteration we would solve partially (3.24) as

$$\begin{aligned} \min_{\alpha_i, i \in B} \quad & \frac{1}{2} \sum_{i \in B} \sum_{j \in B} \alpha_i \alpha_j y_i y_j K_{ij} + \sum_{i \in B} \sum_{j \in \overline{B}} \alpha_i \alpha_j y_i y_j K_{ij} - \sum_{i \in B} \alpha_i \\ s.t. \quad & \begin{cases} 0 \leq \alpha_i \leq C, \;\; i \in B, \\ \sum_{i \in B} \alpha_i y_i = -\sum_{i \in \overline{B}} \alpha_i y_i. \end{cases} \end{aligned} \tag{4.1}$$

Observe that here we have split the quadratic term, discarded $\sum_{i \in \overline{B}} \sum_{j \in \overline{B}} \alpha_i \alpha_j y_i y_j K_{ij}$ for being constant, and applied $\sum_{i \in B} \sum_{j \in \overline{B}} \alpha_i \alpha_j y_i y_j K_{ij} = \sum_{i \in \overline{B}} \sum_{j \in B} \alpha_i \alpha_j y_i y_j K_{ij}$. The constant term $\sum_{i \in \overline{B}} \alpha_i$ is also discarded for being constant. As for the constraint $\sum_{i \in B} \alpha_i y_i = -\sum_{i \in \overline{B}} \alpha_i y_i$, it ensures that globally $\sum_{i=1}^{N} \alpha_i y_i = 0$, which is what we want.

This looks like a very good idea, because subproblem (4.1) is considerably smaller than (3.24), as all the coefficients not in the working set are fixed temporarily. This can also solve the space issues, because we only need to keep the submatrix $K_{Bj} = K_{ij}, i \in B, j = 1, \ldots, N$, of size $|B| \times N$ instead of $N \times N$. The technique is also especially tailored to the problems at hand, making use of the simple constraints and the form of the objective function.

However, there are still some underlying problems: first, subproblem (4.1), although smaller, is still a QP problem that needs to be optimized somehow. It might seem that we did not progress at all; we started from a QP problem we did not know how to solve and ended up with another QP problem that has to be solved as well. Secondly, we should guarantee in some way that the chain of subproblems we are solving means that we solve the global problem. Thirdly, it is possible that solving several smaller subproblems is even slower than solving the global problem.

What was proposed in [56] was to start with a small working set B, solve subproblem (4.1), remove the coefficients that become zero, and add a fixed number M of coefficients

to B. These M coefficients are chosen to be the ones that most violate the KKT optimality conditions (3.25) (we will see later what this means). Then the procedure is repeated sequentially. Eventually all the global support vectors are added to B, so solving (4.1) implies solving the global problem (3.24).

Since we only remove the coefficients that become zero and add a constant number M of coefficients in each iteration, the size of B usually grows linearly. The number of support vectors is also usually proportional to the total number of patterns N, so that the final iterations of chunking involve subproblems of sizes comparable to the size of the full problem. Again, this makes it inapplicable for large datasets. Besides, a specific QP solver for the subproblems has to be used in every iteration.

Later, [57] went a step further and suggested to keep the size of B fixed. In order to do this, it had to be shown that the solution of the global problem can still be attained, provided that at least one coefficient violating the KKT conditions is added to the working set in every iteration. This means that another coefficient has to be dropped from B in every iteration. This procedure was called *decomposition method*, and all algorithms that have followed this approach from that moment on are known as *decomposition algorithms*.

Keeping the size of B at bay, as long as this size is small, potentially solves the memory problems we were mentioning. Now we only need an amount $\mathcal{O}(|B|N)$ of memory, with $|B|$ constant. Besides, convergence to the global optimum is guaranteed. Nevertheless, speed is very dependent on how we choose the example to incorporate to the working set, and how we choose the one to be dropped off. Last but not least, even if the subproblems are equally small, the need of an additional QP solver for solving these subproblems looks like something we cannot get round.

Fortunately, we can address this issue by taking $|B| = 2$. This is the smallest size we can think for the working set, since otherwise we could not satisfy the constraint $\sum_{i\in B} \alpha_i y_i = -\sum_{i\in \overline{B}} \alpha_i y_i$ in (4.1), and the key observation in [26] for devising the SMO algorithm, which is explained in detail next.

4.1.1.1 SMO

As we have seen, the SMO algorithm can be considered as the simplest of all decomposition algorithms, because it makes use of the smallest working set possible, of just two coefficients. The memory requirements are limited thus to $\mathcal{O}(2N)$ space. The main advantage of such a choice, besides simplicity, is that each subproblem of the form (4.1) can be solved analytically, so there is no need of an additional optimization software. This is not the case for any size of the working set greater than two.

Before describing the algorithm, let us fix some notation first. This notation will be used in the sequel with minor changes.

Our objective is to minimize the objective function of (3.24). Because this is the dual problem, we will denote this function as $\mathcal{D}\left(\overrightarrow{\alpha}\right)$, whereas the primal objective function of (3.21) will be written as $\mathcal{P}\left(\overrightarrow{w}, b, \overrightarrow{\xi}\right)$. Since we are solving the task iteratively, we will have a sequence (cf. Definition 2.1) $\left\{\overrightarrow{\alpha}^t\right\}$ of dual candidate solutions, starting from an initial guess $\overrightarrow{\alpha}^0$. If the algorithm is convergent, our hope is that we gradually get closer to an optimum $\overrightarrow{\alpha}^*$.

Even though this optimum $\overrightarrow{\alpha}^*$ might not be unique (if it is not unique, it will be part of a convex set of optima by Theorem 2.50), we know by Theorem 2.63 that the duality gap is zero. This means that, despite the possibility that there are several optima, the optimal value of the dual objective function is unique. The same reasoning applies for the primal objective function. We will denote these values as $\mathcal{D}^*$ and $\mathcal{P}^*$, respectively.

The somewhat loose idea of "getting closer to an optimum" becomes now more clear: by iteratively solving subproblems of the form (4.1), we would like to generate a monotonically decreasing sequence $\left\{\mathcal{D}^t\right\}$ (recall Definition 2.2), where $\mathcal{D}^t$ is the shorthand for $\mathcal{D}\left(\overrightarrow{\alpha}^t\right)$. Moreover, this should be a convergent sequence (Definition 2.5) whose limit is $\mathcal{D}^*$. The study of these sequences is postponed till Chapter 6. In this chapter we will consider just a single iteration.

Let us assume then that we have as the current estimate $\overrightarrow{\alpha}^t$, and we want to calculate the next estimate $\overrightarrow{\alpha}^{t+1}$. SMO needs to select coefficients from $\overrightarrow{\alpha}^t$ and keep the rest of the coefficients fixed. For the sake of simplicity, we will omit the superscript t in the following, although it should be kept in mind that all the quantities mentioned are dependent on the specific iteration we are in.

These two coefficients are usually named L and U. Thus, the update will be of the form

$$\begin{aligned} \alpha_L &\leftarrow \alpha_L + \lambda_L \\ \alpha_U &\leftarrow \alpha_U + \lambda_U \\ \alpha_i &\leftarrow \alpha_i \ \ \forall i \neq L, U, \end{aligned} \tag{4.2}$$

where λ_L and λ_U denote the advance factors. Taking into account the constraint $\sum_{i=1}^{N} \alpha_i y_i = 0$, we must have $y_L \lambda_L + y_U \lambda_U = 0$ and, therefore, $\lambda_U = -y_L y_U \lambda_L$.

Because of (3.23), the quadratic term in the objective function of (3.24) is equal to $(1/2)\|\overrightarrow{w}\|_2^2$. Moreover, the update on $\overrightarrow{w}$ will be of the form

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda_L y_L \Phi(\overrightarrow{x}_L) + \lambda_U y_U \Phi(\overrightarrow{x}_U) = \overrightarrow{w} - \lambda_L y_L \overrightarrow{z}_{UL},$$

where $\overrightarrow{z}_{UL} = \Phi(\overrightarrow{x}_U) - \Phi(\overrightarrow{x}_L)$. Thus, the update in the dual function can be written as a function of λ_L:

$$\begin{aligned}
\mathcal{D} &\leftarrow \frac{1}{2}\|\overrightarrow{w} - \lambda_L y_L \overrightarrow{z}_{UL}\|_2^2 - \sum_{i=1}^{N} \alpha_i - \lambda_L(1 - y_L y_U) \\
&= \frac{1}{2}\|\overrightarrow{w} - \lambda_L y_L \overrightarrow{z}_{UL}\|_2^2 - \sum_{i=1}^{N} \alpha_i + \lambda_L y_L (y_U - y_L) \\
&= \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{\lambda_L^2}{2}\|\overrightarrow{z}_{UL}\|_2^2 - \lambda_L y_L \langle \overrightarrow{w}, \overrightarrow{z}_{UL} \rangle - \sum_{i=1}^{N} \alpha_i + \lambda_L y_L (y_U - y_L) \\
&= \mathcal{D} + \frac{\lambda_L^2}{2}\|\overrightarrow{z}_{UL}\|_2^2 - \lambda_L y_L (\langle \overrightarrow{w}, \overrightarrow{z}_{UL} \rangle - (y_U - y_L)) = \psi(\lambda_L).
\end{aligned}$$

We want as much descent as possible in the dual, so differentiating $\psi(\lambda_L)$ and equaling to zero mean that the unconstrained optimal λ_L is

$$\overline{\lambda}_L = \frac{y_L(\langle \overrightarrow{w}, \overrightarrow{z}_{UL} \rangle - (y_U - y_L))}{\|\overrightarrow{z}_{UL}\|_2^2} = y_L \frac{\Delta}{\|\overrightarrow{z}_{UL}\|_2^2} = y_L \overline{\lambda}, \tag{4.3}$$

where $\Delta = \langle \overrightarrow{w}, \overrightarrow{z}_{UL} \rangle - (y_U - y_L)$ and $\overline{\lambda} = \Delta / \|\overrightarrow{z}_{UL}\|_2^2$. As for λ_U:

$$\overline{\lambda}_U = -y_U y_L \overline{\lambda}_L = -y_U \frac{\Delta}{\|\overrightarrow{z}_{UL}\|_2^2} = -y_U \overline{\lambda}. \tag{4.4}$$

Substituting (4.3) in the dual update implies that

$$\mathcal{D} \leftarrow \mathcal{D} - \frac{[y_L(\langle \overrightarrow{w}, \overrightarrow{z}_{UL} \rangle - (y_U - y_L))]^2}{2\|\overrightarrow{z}_{UL}\|_2^2} = \mathcal{D} - \frac{\Delta^2}{2\|\overrightarrow{z}_{UL}\|_2^2}, \tag{4.5}$$

which in principle guarantees that the sequence $\{\mathcal{D}^t\}$ is monotonically decreasing. However, we must take into account the box constraints that say that α_L and α_U must lie

in $[0, C]$ after the update. Hence, it is possible that the values in (4.3) and (4.4) are too large.

Luckily, optimizing $\psi(\lambda_L)$ is a convex one–dimensional problem, so if the optimal solution is not (4.3), it will be as close to it as possible, as long as the constraints $0 \leq \alpha_L + \lambda_L \leq C$ and $0 \leq \alpha_U + \lambda_U \leq C$ are fulfilled.

Before we can derive the optimal value for λ_L, we have to determine how to select L and U. In his original paper [26], Platt suggested two complex heuristics to perform this task. Basically, the first heuristic is an outer loop that looks for an L such that the pattern $\overrightarrow{x}_L$ violates the KKT conditions (3.25), whereas the second heuristic is an inner loop that looks for an U that maximizes the dual change (4.5), once L is selected. For more details, please refer to [58].

It soon became evident that these heuristics were too complex. Keerthi *et al.* [59] derived a much simpler selection scheme, which is currently known in the literature as *first order selection*. This selection is a very natural one, considering what we have already explained. First, it should be noted that the gradient of the dual objective function, denoted by $\nabla\mathcal{D}$, is

$$\nabla\mathcal{D} \quad = \quad \Omega\overrightarrow{\alpha} - \overrightarrow{1}, \tag{4.6}$$

where Ω is the matrix with elements $\Omega_{ij} = y_i y_j K_{ij}$. Then the i–th element of this gradient is given by

$$\nabla\mathcal{D}_i \quad = \quad (\Omega\overrightarrow{\alpha})_i - 1 = y_i \sum_{j=1}^{N} \alpha_j y_j K_{ij} - 1 = y_i \langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle - 1,$$

so Δ can be rewritten as

$$\Delta \quad = \quad \langle \overrightarrow{w}, \overrightarrow{z}_{UL} \rangle - (y_U - y_L) = y_U \nabla\mathcal{D}_U - y_L \nabla\mathcal{D}_L. \tag{4.7}$$

Looking back at the dual gain in (4.5), we are interested in maximizing this gain, so as to advance as quickly as possible in the dual. However, maximizing the fraction $\Delta / \|\overrightarrow{z}_{UL}\|_2$ would imply checking all possible (L, U) pairs, which has a cost of $\mathcal{O}(N^2)$.

Recall that this cost is unacceptable, because $\mathcal{O}(N^2)$ per iteration would mean $\mathcal{O}(N^3)$ in total, once the number of iterations are dependent on the number of patterns.

For this reason, first order selection in [59] does not try to maximize exactly this fraction, but to do so approximately. Ignoring the denominator, we would like to maximize the numerator, that is, Δ. Using (4.7), the scheme is in principle

$$U = \arg\max_i \{y_i \nabla \mathcal{D}_i\}, \qquad L = \arg\min_i \{y_i \nabla \mathcal{D}_i\}, \tag{4.8}$$

so that Δ is non–negative and as large as possible. Nevertheless, the box constraints should be taken into account. By virtue of (4.3), if $y_L = +1$ then $\lambda_L > 0$, so it must be the case that $\alpha_L < C$. Similarly, if $y_L = -1$ we must have $\alpha_L > 0$. The analysis for λ_U is analogous in view of (4.4): if $y_U = +1$ then $\lambda_U < 0$, so it is required that $\alpha_U > 0$, whereas if $y_U = -1$ we should have $\alpha_U < C$, because $\lambda_U > 0$. As a result, (4.8) must be refined to

$$L = \arg\min_{i \in \mathcal{I}_L} \{y_i \nabla \mathcal{D}_i\}, \qquad U = \arg\max_{i \in \mathcal{I}_U} \{y_i \nabla \mathcal{D}_i\}, \tag{4.9}$$

where we introduce the index sets

$$\begin{array}{rcl} \mathcal{I}_L & = & \{i : (y_i = +1 \wedge \alpha_i < C) \vee (y_i = -1 \wedge \alpha_i > 0)\}, \\ \mathcal{I}_U & = & \{i : (y_i = +1 \wedge \alpha_i > 0) \vee (y_i = -1 \wedge \alpha_i < C)\}. \end{array} \tag{4.10}$$

Assuming that the $y_i \nabla \mathcal{D}_i$ values are stored in memory, first order selection has a cost of $\mathcal{O}(1)$ operations. In contrast, carrying out a full search over the whole set of possible pairs (L, U) has an $\mathcal{O}(N^2)$ cost. Another possibility, introduced by Fan *et al.* in [60] and also studied by Glasmachers *et al.* in [61], is to arrive to a compromise between these two costs. This selection, termed *second order selection*, can be summarized in the following way:

$$L = \arg\min_{i \in \mathcal{I}_L} \{y_i \nabla \mathcal{D}_i\}, \qquad U = \arg\max_{i \in \mathcal{I}_U, y_i \nabla \mathcal{D}_i > y_L \nabla \mathcal{D}_L} \left\{ \frac{(y_i \nabla \mathcal{D}_i - y_L \nabla \mathcal{D}_L)^2}{\| \overrightarrow{z}_{iL} \|_2^2} \right\}. \tag{4.11}$$

What this selection scheme does is to select L exactly as in first order selection. Once L is selected, the selection of U seeks to maximize the dual change of (4.5), this time

Algorithm 1 SMO for l_1–SVC

while stopping criterion not true **do**
  Select working set (L, U) with first order (Eq. (4.9)) or second order (Eq. (4.11)).
  Compute $\overline{\lambda} = \Delta / \| \overrightarrow{z}_{UL} \|_2^2$.
  Clip $\overline{\lambda}$ if necessary using (4.12).
  Compute λ_L and λ_U with (4.3) and (4.4).
  Update $\overrightarrow{\alpha}$ with (4.2).
end while

considering the denominator as well. The numerator is ensured to remain positive. Since every possible choice for U requires the calculation of $\| \overrightarrow{z}_{iL} \|_2^2$, it is seen that second order selection has a cost of $\mathcal{O}(N)$ per iteration.

Whether we choose the working set with first order or with second order selection, it is guaranteed that $\Delta > 0$. It remains to see which is the optimal advance factor when $\overline{\lambda}$ provokes that we get out of bounds. By a similar argument to the one used for the index sets $\mathcal{I}_U$ and $\mathcal{I}_L$, four different clips are possible:

$$\begin{array}{rcll} \overline{\lambda} & \leftarrow & \min\{\overline{\lambda}, \alpha_L\} & \text{if} \quad y_L = -1, \\ \overline{\lambda} & \leftarrow & \min\{\overline{\lambda}, C - \alpha_L\} & \text{if} \quad y_L = +1, \\ \overline{\lambda} & \leftarrow & \min\{\overline{\lambda}, C - \alpha_U\} & \text{if} \quad y_U = -1, \\ \overline{\lambda} & \leftarrow & \min\{\overline{\lambda}, \alpha_U\} & \text{if} \quad y_U = +1. \end{array} \tag{4.12}$$

Once we have applied (4.12), it is readily seen that applying (4.3) and (4.4) is safe and fulfills the box constraints. In this way, we can summarize the SMO algorithm for l_1–SVC in Algorithm 1. The only unclear point is the stopping criterion: we will cover this topic in depth in Chapter 5, where it will be seen how the violation of the KKT conditions relates to the working set selection, and can also be used very naturally as a stopping condition.

In practice, second order works better than first order, because this additional cost is usually more than compensated, as the number of iterations is substantially reduced. This is why this selection is the one performed by LIBSVM [62], which can be considered as the *de facto* standard for training non–linear SVMs. This software, besides second order selection, uses some computational tricks and efficiencies to be faster. Among them, the most important ones are *caching* and *shrinking*.

Caching is aimed to minimize the number of kernel operations during the algorithm. Because of (4.11), we should ideally have the L–th row of the kernel matrix in memory in every iteration, so that it is not necessary to recalculate it. However, recall that usually, especially with large datasets, we will not be able to have the full kernel matrix in memory. An intelligent scheme to decide which rows are disposable is therefore

advisable. Basically speaking, what LIBSVM does is to keep in memory as many rows of the kernel matrix as possible, and replace them with a Least Recently Used (LRU) scheme. This means that, whenever a row not already in memory has to be calculated, it replaces the row that has not been used for a longest time. Building on this idea of keeping a kernel cache, some other working set selections have been proposed, most notably [63], where a scheme is suggested that combines the idea of maximizing the dual gain, yet at the same time trying to use kernel rows which are already in the cache.

This thesis ignores the effects of caching, as it is a computational issue with no influence on the theoretical aspect. For a detailed study on the subject, a valuable reference is [64]. As for shrinking, it will be briefly described and justified in Chapter 6, once we establish the algorithm's convergence. For the time being, it suffices to know that it is a heuristic also aimed to minimize the computational burden.

Even though SMO has been described here for the case of l_1–SVC, there are different versions of the algorithm for the different SVM formulations of Chapter 3. Different selection schemes have also been studied:

- For LS–SVC and LS–SVR, the main reference is [65], where a first order scheme was developed in the lines of [59]. Second order selection has been treated in [66] and, more recently, by the author in [67].
- For ϵ–insensitive SVR, the decomposition problem of the form (4.1) is more complex. Recalling the dual formulation (3.50), now there are two coefficient vectors $\overrightarrow{\alpha}$ and $\overrightarrow{\alpha}'$. This caused that the first adaptations of SMO to SVR selected four coefficients instead of two in [68]. Shevade *et al.* suggested again a first order selection in [69]. Flake *et al.* [70] adopted an alternative approach where they used variables $\lambda_i = \alpha_i - \alpha'_i$: this causes that the objective function has a term with absolute value, which is only piece–wise differentiable and presents some difficulties. Not until the publication of [71] was the possibility of selecting just two coefficients considered.
- For ν–SVC, an SMO version was suggested by Chang *et al.* in [72]. The same authors extended their ideas for ν–SVR in [48].

References [71], [72] and [48] were all written by Chih–Jen Lin and his coworkers. From the publication of [71] onwards, this team has formulated SVC, SVR, OC, ν–SVC and ν–SVR under a common framework, making use of a generalized dual formulation that encompasses all these problems. This is precisely the formulation used internally in LIBSVM, which was also implemented by this team.

Building on their work, we will give in Chapter 5 an even more general formulation, both in the primal and in the dual, that includes these formulations, as well as LS–SVMs and some other SVM–related formulations, such as the geometric ones described in §3.4. Furthermore, this very general formulation will also allow us to devise an SMO algorithm for it that includes as particular cases the separate SMO algorithms for every particular case.

Next, we describe briefly some other decomposition algorithms, as well as some other SVM algorithms that do not rely on decomposition ideas.

4.1.1.2 Other Decomposition Algorithms

We have seen that restricting the working set size to two in SMO has the advantage that the resulting subproblem is solvable analytically. This is not the case when the size is larger, but it can be argued that, if we select the working set carefully enough, faster progress in the dual is possible. If this progress is much faster than in SMO, it may compensate the additional overhead that each iteration requires for optimizing the resulting subproblem.

This is the idea behind SVM^{light}, an algorithm invented by Joachims in [73] for l_1–SVC, which stands out as the most prominent decomposition algorithm after SMO. It allows any even size q for the working set. In order to select q coefficients for the working set, Zoutendijk's method [74] is employed. Specifically, SVM^{light}'s working set selection is performed by solving the optimization problem

$$\begin{array}{ll} \min\limits_{\overrightarrow{d}} & \left\langle \nabla\mathcal{D}, \overrightarrow{d} \right\rangle \\ s.t. & \begin{cases} \left\langle \overrightarrow{d}, \overrightarrow{y} \right\rangle = 0, & \\ d_i \geq 0, & \forall i : \alpha_i = 0, \\ d_i \leq 0, & \forall i : \alpha_i = C, \\ -\overrightarrow{1} \leq \overrightarrow{d} \leq \overrightarrow{1}, & \\ \| \overrightarrow{d} \|_0 = q, & \end{cases} \end{array} \tag{4.13}$$

where $\nabla\mathcal{D}$ is given by (4.6). The direction $\overrightarrow{d}$ determines which coefficients are to be fixed in the current iteration (those with $d_i = 0$), and which coefficients enter in the optimization subproblem of the form (4.1) (those with $d_i \neq 0$). The last constraint sets

the size of this optimization subproblem to q, as exactly q elements are required to be non–zero.

The objective function of the above problem seeks to maximize the change of the dual, considering that the dual objective function can be approximated by a first–order Taylor polinomial. Specifically, we have that $\mathcal{D}(\overrightarrow{\alpha} + \lambda \overrightarrow{d}) \approx \mathcal{D}(\overrightarrow{\alpha}) + \lambda \langle \nabla \mathcal{D}, \overrightarrow{d} \rangle$. Since $\langle \nabla \mathcal{D}, \overrightarrow{d} \rangle < 0$ for any descent direction, minimizing this quantity looks for the approximately steepest descent direction with q non–zero elements.

The constraint $\left\langle \overrightarrow{d}, \overrightarrow{y} \right\rangle = 0$ ensures that $\sum_{i=1}^{N} (\alpha_i + \lambda d_i) y_i = 0$, whereas the constraints $d_i \geq 0, \forall i : \alpha_i = 0$ and $d_i \leq 0, \forall i : \alpha_i = C$ make sure that $0 \leq \alpha_i + \lambda_i d_i \leq C \ \forall i$. Thus, the chosen direction will not only be a descent one, but also feasible for small enough values of λ. Finally, the constraint $-\overrightarrow{1} \leq \overrightarrow{d} \leq \overrightarrow{1}$ bounds the possible values of $\overrightarrow{d}$. Otherwise the problem is unbounded: once we find a $\overrightarrow{d}$ such that $\langle \nabla \mathcal{D}, \overrightarrow{d} \rangle < 0$, this quantity could be made arbitrarily small by scaling the direction with an arbitrarily large positive factor.

Even if problem (4.13) looks like a difficult task to solve, especially for the non–convex constraint of having exactly q non–zero elements, Joachims realized that it is easily solvable. The procedure is the following:

1. Set $\overrightarrow{d} = \overrightarrow{0}$.
2. Sort the coefficients α_i in decreasing order according to the values $y_i \nabla \mathcal{D}_i$.
3. Assign $d_i = -y_i$ for the first $q/2$ coefficients in the sorted list such that $d_i = -y_i$ is feasible.
4. Assign $d_i = +y_i$ for the last $q/2$ coefficients in the sorted list such that $d_i = +y_i$ is feasible.

For details on why this is so, we refer the reader to [58, 73]. As a result, the cost of solving (4.13) is $\mathcal{O}(N \log N)$, which is the best that can be done for sorting the N values $y_i \nabla \mathcal{D}_i$.

Alongside this computational overhead of sorting, we have to add the fact of invoking a QP solver for optimizing the resulting subproblem. In SVM^{light} there are two possible solvers at hand: MINOS [75] and LOQO [76]. Whichever the choice, the overall cost of SVM^{light} is reported to be between $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ [73]. This was far better than the complexity of chunking, which remained in $\mathcal{O}(N^3)$ levels, but Joachims observed that the optimal size of the working set was usually $q = 2$.

It should come as no surprise that the case $q = 2$ reduces to SMO with first order selection (although SVM^{light} is slower in this case because it does not solve the subproblems analytically). This was shown first in [60], and with somewhat more detail in [58].

Therefore, SVM^{light} can be seen as a generalization of first order SMO that allows for any even size for the working set. In fact, SVM^{light}'s implementation also makes use of caching and shrinking. Unfortunately, there is no straightforward way to generalize second order SMO for larger working sets, since no problem of the form (4.13) with a quadratic term can be solved analytically. This is the reason why in practice LIBSVM, which applies second order selection, has superseded SVM^{light} as the standard software for SVM training.

After its appearance, some modifications of SVM^{light} were suggested. An important example is the one of Palagi *et al.* [77], where a change in the subproblems being solved facilitates the proof of convergence for the algorithm (this topic will be covered in detail in Chapter 6, which is devoted to convergence). Another algorithm suggested by the same team is [78]: here an alternative selection scheme cycles through all the coefficients in such a way that convergence is also ensured.

SVM^{light} was also extended to cover the l_1–SVR problem, giving rise to the SVMTorch method [79]. A very similar problem to (4.13), also analytically solvable, can be posed for the working set selection. In any case, this software has also been displaced by LIBSVM, as it also covers SVR and often behaves better.

We move next to some other approaches for SVM training not based on decomposition.

4.1.2 Other SVM algorithms

The dual formulations have been recently studied almost exclusively under a decomposition perspective. Some attempts not in this category are the so–called *interior point methods* [80]. Even though these algorithms are tailored to the specific structure of the dual, they suffer from too large time and memory requirements to be practical in large problems.

It is in the primal formulations that research has been concentrated lately. With the advent of massive amounts of data, there are a number of applications, such as text mining or web mining, where data are often very high dimensional, so the use of a kernel proves to be quite redundant. Using the standard dot product means that the primal formulations are no longer intractable, and a number of algorithms specifically tailored for the primal have been proposed, again concentrating specially on the l_1–SVC case.

For instance, using the fact that the hinge loss is piecewise differentiable, the Pegasos algorithm follows a projected subgradient strategy [81]. Other methods are based on the cutting–plane algorithm, the most prominent examples being OCAS [82] and SVM-Perf [83]. An interesting method that combines optimization in the dual while keeping stored the weight vector $\overrightarrow{w}$ in memory is LIBLINEAR [84]. However, this last method considers an SVM formulation with no bias term, something which is known in the literature as *homogeneous SVMs*.

As we outlined previously, such algorithms are beyond the scope of this thesis. We will pay special attention to non–linear SVMs, which preclude the use of all these methods, and study the SMO algorithm and the geometric algorithms described next in detail.

4.2 Geometric Algorithms

In this section we will analyze two algorithms, namely the Gilbert–Schlesinger–Kozinec (GSK) and the Mitchell–Dem'yanov–Malozemov (MDM) methods. We shall begin with the simplest geometric problem from Chapter 3, that is, the Minimum Norm Problem in standard convex hulls. Next, we will see how extending these methods for the Nearest Point Problem is quite straightforward, since the point in a hull can be considered as the origin of coordinates for the points lying in the other hull. Finally, we will study the case with reduced convex hulls, this time directly for RCH–NPP. While the straight generalization of GSK works, the same cannot be said for MDM. It will not be until Chapter 5 that a satisfactory solution is obtained.

The notation used in the descriptions follows the lines of the one used in §4.1.1.1 for SMO. Once more, the subscript t is omitted for the sake of clarity, except in the figures, to distinguish the current estimate from the next one.

4.2.1 Algorithms for CH–MNP

The problem of finding the nearest point of a polytope to the center of coordinates was studied much before the invention of Support Vector Machines. In fact, the GSK algorithm stems from Gilbert's method [85], which was devised in the 60s. The MDM algorithm [28] was developed some years later, in the 70s.

4.2.1.1 Gilbert's Algorithm

Gilbert's algorithm uses a single pattern $\Phi(\overrightarrow{x}_L)$ to update the current weight vector $\overrightarrow{w}$ in the form

$$\overrightarrow{w} \leftarrow (1-\lambda)\overrightarrow{w} + \lambda\Phi(\overrightarrow{x}_L),$$

that is, it builds the convex combination of the weight vector and the selected point. This can be rewritten as

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda(\Phi(\overrightarrow{x}_L) - \overrightarrow{w}) = \overrightarrow{w} + \lambda\overrightarrow{z}_L,$$

where we define $\overrightarrow{z}_L = \Phi(\overrightarrow{x}_L) - \overrightarrow{w}$. Since $\overrightarrow{w}$ is itself influencing in the update, all the coefficients will be updated, so that

$$\begin{array}{rcl} \alpha_L & \leftarrow & (1-\lambda)\alpha_L + \lambda \\ \alpha_i & \leftarrow & (1-\lambda)\alpha_i \;\; \forall i \neq L. \end{array} \tag{4.14}$$

Analogously to SMO, we can express the change in the dual (3.68) as a function of λ:

$$\begin{array}{rcl} \mathcal{D} & \leftarrow & \frac{1}{2}\|\overrightarrow{w} + \lambda\overrightarrow{z}_L\|_2^2 \\ & = & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{\lambda^2}{2}\|\overrightarrow{z}_L\|_2^2 + \lambda\langle\overrightarrow{w}, \overrightarrow{z}_L\rangle \\ & = & \mathcal{D} + \frac{\lambda^2}{2}\|\overrightarrow{z}_L\|_2^2 + \lambda\langle\overrightarrow{w}, \overrightarrow{z}_L\rangle = \psi(\lambda). \end{array}$$

We want as much descent as possible in the dual. Differentiating $\psi(\lambda)$ and equaling to zero yield the unconstrained optimal λ:

$$\lambda = -\frac{\langle\overrightarrow{w}, \overrightarrow{z}_L\rangle}{\|\overrightarrow{z}_L\|_2^2} = -\frac{\Delta}{\|\overrightarrow{z}_L\|_2^2}, \tag{4.15}$$

where $\Delta = \langle\overrightarrow{w}, \overrightarrow{z}_L\rangle$. Substituting (4.15) in the dual update gives

$$\mathcal{D} \leftarrow \mathcal{D} - \frac{\Delta^2}{2\|\overrightarrow{z}_L\|_2^2}. \tag{4.16}$$

Now we should decide how to choose L. In order to obtain a descent direction, it should hold that $\psi'(0) < 0$. Thus, $\langle \overrightarrow{w}, \overrightarrow{z}_L \rangle < 0$, so $\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_L) \rangle < \langle \overrightarrow{w}, \overrightarrow{w} \rangle = \|\overrightarrow{w}\|_2^2$. This can be guaranteed if we choose L as

$$L \quad = \quad \arg\min_i \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}. \tag{4.17}$$

Observe that such a choice minimizes Δ and makes it negative, so $\lambda > 0$ in (4.15), and the unconstrained dual change in (4.16) is approximately maximized (ignoring the denominator, as first order SMO did). Hence, Gilbert's algorithm can be considered in a sense as another first order method.

The unconstrained optimal value (4.15) may cause that we get out of the convex hull of the samples when applying the update. Clipping this value is much simpler than in SMO: since we are optimizing on the line segment between $\overrightarrow{w}$ and $\Phi(\overrightarrow{x}_L)$, any point in this segment is inside the hull, because it is a convex set (recall Definition 2.40). Since $\lambda > 0$, the only possibility is that it becomes too large, so the optimal λ is then

$$\lambda \quad \leftarrow \quad \min\{\lambda, 1\}. \tag{4.18}$$

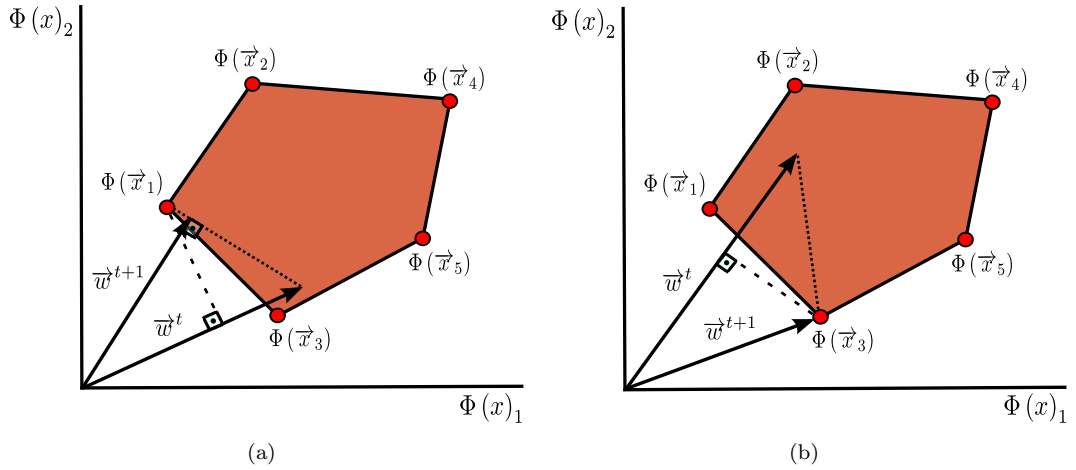

FIGURE 4.1: Example of two iterations of Gilbert's algorithm: 4.1(a) $L = 1$ and the optimal λ need not be clipped, because the closest point to the origin lies inside the segment joining w^t and $\Phi(x_1)$, 4.1(b) $L = 3$ and the optimal λ has to be clipped to 1, because the closest point to the origin is precisely $\Phi(x_3)$.

Examples of these two situations are given in Figure 4.1. When the closest point to the origin in the segment $[\overrightarrow{w}, \Phi(\overrightarrow{x}_L)]$ is $\Phi(\overrightarrow{x}_L)$ itself, then $\lambda = 1$. If this is not so, but it is a point in the middle of the segment, then $\lambda = -\Delta / \|\overrightarrow{z}_L\|_2^2$. It is also illustrated how L

Algorithm 2 Gilbert's algorithm for CH–MNP

while $\Delta > \epsilon$ **do**
  Select L with (4.17).
  Compute the unconstrained λ with (4.15).
  Clip λ if necessary using (4.18).
  Update $\overrightarrow{\alpha}$ with (4.14).
end while

is the index belonging with the point with the smallest "margin". In this case, "margin" has to be understood as projection along the vector $\overrightarrow{w}$.

The stopping criterion in Gilbert's algorithm is also simple. Because of Theorem 2.51, the minimum of $\langle \overrightarrow{w}, \Phi(\overrightarrow{x}) \rangle$ is attained in an extreme point, that is, in $\Phi(\overrightarrow{x}_L)$. Besides, looking back at Figure 3.8, it is clear that for the optimal $\overrightarrow{w}^*$ we get $\langle \overrightarrow{w}^*, \Phi(\overrightarrow{x}_L) \rangle = \langle \overrightarrow{w}^*, \overrightarrow{w}^* \rangle$. That is, at the optimal $\overrightarrow{w}^*$ we have $\Delta^* = 0$, whereas for any suboptimal $\overrightarrow{w}$ we have $\Delta > 0$. Thus, it seems natural to stop when $\Delta < \epsilon$, with $\epsilon > 0$ a small value.

Gilbert's algorithm is summarized in Algorithm 2.

4.2.1.2 Mitchell–Dem'yanov–Malozemov (MDM) Algorithm

Gilbert's algorithm selects a single pattern in every iteration for updating. The MDM algorithm uses instead two patterns $\Phi(\overrightarrow{x}_L)$ and $\Phi(\overrightarrow{x}_U)$ to update $\overrightarrow{w}$ as

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda_L \Phi(\overrightarrow{x}_L) + \lambda_U \Phi(\overrightarrow{x}_U).$$

In terms of the α_i coefficients the update is

$$\begin{aligned} \alpha_L &\leftarrow \alpha_L + \lambda_L, \\ \alpha_U &\leftarrow \alpha_U + \lambda_U, \\ \alpha_i &\leftarrow \alpha_i \ \ \forall i \neq L, U. \end{aligned} \tag{4.19}$$

Because of the constraints of (3.68), the sum of these coefficients should be still 1, so $\lambda_L = -\lambda_U$. Then, the update on $\overrightarrow{w}$ can be rewritten as a function of λ_L:

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda_L(\Phi(\overrightarrow{x}_L) - \Phi(\overrightarrow{x}_U)) = \overrightarrow{w} + \lambda_L \overrightarrow{z}_{LU},$$

where we define $\overrightarrow{z}_{LU} = \Phi(\overrightarrow{x}_L) - \Phi(\overrightarrow{x}_U)$.

The change in the dual (3.68) is also expressible as a function of λ_L:

$$\begin{aligned}\mathcal{D} &\leftarrow \frac{1}{2}\|\overrightarrow{w} + \lambda_L \overrightarrow{z}_{LU}\|_2^2 \\ &= \frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{\lambda_L^2}{2}\|\overrightarrow{z}_{LU}\|_2^2 + \lambda_L\langle\overrightarrow{w}, \overrightarrow{z}_{LU}\rangle \\ &= \mathcal{D} + \frac{\lambda_L^2}{2}\|\overrightarrow{z}_{LU}\|_2^2 + \lambda_L\langle\overrightarrow{w}, \overrightarrow{z}_{LU}\rangle = \psi(\lambda_L).\end{aligned}$$

Differentiating $\psi(\lambda_L)$ and equaling to zero yield the unconstrained optimum

$$\lambda_L = -\frac{\langle\overrightarrow{w}, \overrightarrow{z}_{LU}\rangle}{\|\overrightarrow{z}_{LU}\|_2^2} = -\frac{\Delta}{\|\overrightarrow{z}_{LU}\|_2^2}, \tag{4.20}$$

defining $\Delta = \langle\overrightarrow{w}, \overrightarrow{z}_{LU}\rangle$. Substitution of (4.15) in the dual update gives

$$\mathcal{D} \leftarrow \mathcal{D} - \frac{\Delta^2}{2\|\overrightarrow{z}_{LU}\|_2^2}. \tag{4.21}$$

It remains to choose L and U. As with Gilbert's algorithm, a descent direction has $\psi'(0) < 0$. In this case, this implies $\langle\overrightarrow{w}, \overrightarrow{z}_{LU}\rangle < 0$, so $\langle\overrightarrow{w}, \Phi(\overrightarrow{x}_L)\rangle < \langle\overrightarrow{w}, \Phi(\overrightarrow{x}_U)\rangle$. This can be guaranteed if we choose in principle

$$L = \arg\min_i\{\langle\overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}, \qquad U = \arg\max_i\{\langle\overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}.$$

This choice minimizes Δ and makes it negative, so $\lambda_L > 0$ in (4.20). In turn, $\lambda_U = -\lambda_L < 0$. The coefficients must remain non–negative, so α_U should be positive to ensure that after the update it remains non–negative. There is no problem with L, because the constraint $\sum_{i=1}^N \alpha_i = 1$ already ensures that $\alpha_i \leq 1\ \forall i$, so we cannot get out of the hull. Hence, the final selection is

$$L = \arg\min_i\{\langle\overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}, \qquad U = \arg\max_{i:\alpha_i>0}\{\langle\overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}. \tag{4.22}$$

The above reasoning also tells us that λ_L need not be clipped from above. However, it may have to be clipped from below via

$$\lambda_L \leftarrow \min\{\lambda_L, \alpha_U\}. \tag{4.23}$$

Examples of these two situations are given in Figure 4.2. In the first case, $\overrightarrow{w}$ is assumed to be the midpoint between $\Phi(\overrightarrow{x}_2)$ and $\Phi(\overrightarrow{x}_3)$, which are the only points with non–zero α_i (specifically, $\alpha_2 = \alpha_3 = 1/2$). Then the algorithm chooses $U = 2$, $L = 3$ and moves towards $\Phi(\overrightarrow{x}_3)$, until it is reached by making $\lambda_L = \alpha_U$, so that $\alpha_3 = 1$ and $\alpha_2 = 0$. In the next iteration $U = 3$, $L = 1$, so we move towards $\Phi(\overrightarrow{x}_1)$. However, we do not reach it, because the point with minimum norm is in between. This happens to be the global optimum of the full problem, as was depicted in Figure 3.8. As in Gilbert's method, L is the index belonging with the point with the smallest projection along $\overrightarrow{w}$, whereas U corresponds to the point which is a support vector and which has the largest projection.

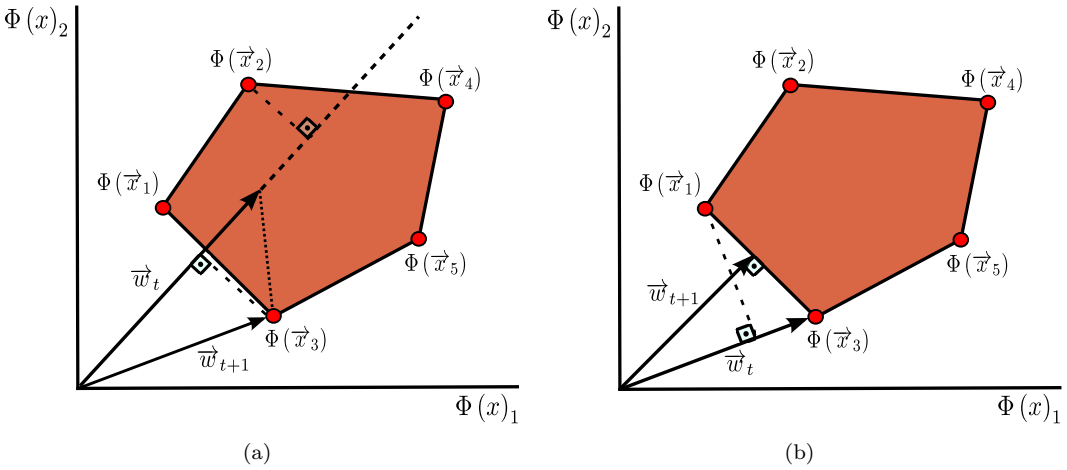

FIGURE 4.2: Example of two iterations of MDM algorithm: 4.2(a) $U = 2$, $L = 3$ and the optimal λ_L is clipped to α_U, arriving thus to $\Phi(x_3)$, 4.2(b) $U = 3$, $L = 1$ and the optimal λ_L need not be clipped, arriving to the global optimum.

The stopping criterion in the MDM algorithm is exactly the same as in Gilbert's case. As a matter of fact, we have again $\Delta^* = 0$. This happens because $\overrightarrow{w}^*$ is either an extreme point of the convex hull or a combination of the extreme points that delimit the face of the hull closest to the origin (see again Figure 3.8). Hence, once in $\overrightarrow{w}^*$, the only points with $\alpha_i^* > 0$ are exactly those in the closest face, or the single optimum itself, and it holds that $\langle \overrightarrow{w}^*, \overrightarrow{x}_L \rangle = \langle \overrightarrow{w}^*, \overrightarrow{x}_U \rangle = \langle \overrightarrow{w}^*, \overrightarrow{w}^* \rangle$.

MDM's algorithm is summarized in Algorithm 3.

Algorithm 3 MDM's algorithm for CH–MNP

while $\Delta > \epsilon$ **do**
  Select L and U with (4.22).
  Compute the unconstrained λ with (4.20).
  Clip λ if necessary using (4.23).
  Update $\overrightarrow{\alpha}$ with (4.19).
end while

4.2.2 Algorithms for CH–NPP

Once we know how to determine the closest point of a convex hull to the origin, a natural extension is to determine the two closest points in two convex hulls. Both the Gilbert and the MDM algorithms have been extended to cover the CH–NPP formulation. This is quite straightforward, because if, say, we are updating the point in the convex hull generated by the positive class, we can take the point in the convex hull of the negative class as the origin of coordinates.

This will become clear next, as we describe the extension of Gilbert's and MDM algorithms with the same convention than the original ones.

4.2.2.1 Gilbert–Schlesinger–Kozinec (GSK) Algorithm

As explained in §3.4.2, in NPP the vector $\overrightarrow{w}$ is decomposed into two components $\overrightarrow{w}_+$ and $\overrightarrow{w}_-$: $\overrightarrow{w}_+$ lying in the hull of the positive class, and $\overrightarrow{w}_-$ in the negative one. Only one of the components is updated in an iteration, so it has to be decided which class we will update on. When updating on the positive class, the origin of coordinates will be $\overrightarrow{w}_-$, and when updating on the negative one it will be $\overrightarrow{w}_+$. To simplify the notation we will write a subscript y to indicate $+$ or $-$.

The extension of Gilbert's algorithm for this case was implemented by Franc *et al.* in [27], building on previous work by M.I. Schlesinger and B.N. Kozinec, hence the name GSK. They suggest to update $\overrightarrow{w}$ as

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda y_L(\Phi(\overrightarrow{x}_L) - \overrightarrow{w}_{y_L}) = \overrightarrow{w} + \lambda y_L \overrightarrow{z}_L,$$

with $\overrightarrow{z}_L = \Phi(\overrightarrow{x}_L) - \overrightarrow{w}_{y_L}$, and y_L stands for the class label of the instance $\overrightarrow{x}_L$. The above equation is valid whether $y_L = +1$ or $y_L = -1$, and has a very similar form to the one for the MNP case. The only change is in the definition of $\overrightarrow{z}_L$ and the apparition of y_L.

The update in the coefficients now changes from (4.14) to

$$\begin{array}{rcl} \alpha_L & \leftarrow & (1-\lambda)\alpha_L + \lambda, \\ \alpha_i & \leftarrow & (1-\lambda)\alpha_i \;\; \forall i : (i \neq L) \wedge (y_i = y_L), \\ \alpha_i & \leftarrow & \alpha_i \;\; \forall i : y_i \neq y_L, \end{array} \tag{4.24}$$

so that the coefficients of the other hull remain the same. This indicates that we are optimizing on the line segment joining $\overrightarrow{w}_{y_L}$ with $\Phi(\overrightarrow{x}_L)$. The update in the dual is seen to be

$$\mathcal{D} \;=\; \mathcal{D} + \frac{\lambda^2}{2}\|\overrightarrow{z}_L\|_2^2 + \lambda y_L \langle \overrightarrow{w}, \overrightarrow{z}_L \rangle = \psi(\lambda).$$

As for the unconstrained optimal λ:

$$\lambda \;=\; -y_L \frac{\langle \overrightarrow{w}, \overrightarrow{z}_L \rangle}{\|\overrightarrow{z}_L\|_2^2} = -y_L \frac{\Delta_{y_L}}{\|\overrightarrow{z}_L\|_2^2}, \tag{4.25}$$

where $\Delta_{y_L} = \langle \overrightarrow{w}, \overrightarrow{z}_L \rangle$ (the subscript y_L is added to reinforce the idea that Δ depends on the class chosen). The dual update if no clipping is needed is then

$$\mathcal{D} \;\leftarrow\; \mathcal{D} - \frac{\Delta_{y_L}^2}{2\|\overrightarrow{z}_L\|_2^2}. \tag{4.26}$$

To obtain a descent direction, it should hold now that $y_L \Delta_{y_L} < 0$, so that (4.25) becomes positive. Then, if $y_L = +1$ we should have $\Delta_{y_L} = \Delta_+ < 0$, whereas if $y_L = -1$ we ought to ensure that $\Delta_{y_L} = \Delta_- > 0$. This can be fulfilled by choosing

$$\begin{array}{rcl} L_+ & = & \arg\min_{i \in \mathcal{I}_+} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \\ L_- & = & \arg\max_{i \in \mathcal{I}_-} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \end{array} \tag{4.27}$$

which respectively minimize and maximize Δ. The class finally chosen is the one for which $-y_L \Delta_{y_L}$ is largest, that is, the one that provides greatest descent. Once the class is selected, λ is clipped in the same way that in Gilbert's method, because we are optimizing again on a line segment:

Algorithm 4 GSK algorithm for CH–NPP

while $-y_L\Delta_{y_L} > \epsilon$ **do**
  Choose L_+ and L_- with (4.27).
  Select the class for which $-y_L\Delta_{y_L}$ is largest.
  Compute the unconstrained λ with (4.25).
  Clip λ if necessary using (4.28).
  Update $\overrightarrow{\alpha}$ with (4.24).
end while

$$\lambda \leftarrow \min\{\lambda, 1\}. \tag{4.28}$$

It is not difficult to see, by generalizing the reasoning for CH–MNP, that $\Delta^*_+ = 0 = \Delta^*_-$, so that $y_L\Delta_{y_L} = 0$ and no descent direction can be found. Therefore, operating analogously to Gilbert's algorithm, GSK stops when $-y_L\Delta_{y_L} \leq \epsilon$.

The pseudocode for the GSK algorithm can be seen in Algorithm 4. Note that the only additional step with respect to Gilbert's method is the class selection.

4.2.2.2 MDM Algorithm

The MDM algorithm was extended for the CH–NPP formulation by Keerthi *et al.* in [86]. Although they termed it the Nearest Point Algorithm (NPA), we stick to the name MDM, because, strictly speaking, any algorithm that solves NPP can be called NPA.

As for MNP, we have to select two indices U and L. Since we are operating with two classes, it has to be decided as in GSK which hull we will optimize on. The update on $\overrightarrow{w}$ is

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda_L y_L \Phi(\overrightarrow{x}_L) + \lambda_U y_U \Phi(\overrightarrow{x}_U).$$

In terms of the α_i coefficients the update is exactly the same as for standard MDM:

$$\begin{aligned} \alpha_L &\leftarrow \alpha_L + \lambda_L, \\ \alpha_U &\leftarrow \alpha_U + \lambda_U, \\ \alpha_i &\leftarrow \alpha_i \ \ \forall i \neq L, U. \end{aligned} \tag{4.29}$$

The constraints of (3.70) tell us that, whichever class we choose, the sum of its coefficients should remain equal to 1, so again $\lambda_L = -\lambda_U$. This, together with the requirement that $y_L = y_U$, allows us to re-express the update on $\overrightarrow{w}$ as a function of λ_L:

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda_L y_L(\Phi(\overrightarrow{x}_L) - \Phi(\overrightarrow{x}_U)) = \overrightarrow{w} + \lambda_L y_L \overrightarrow{z}_{LU},$$

where we define $\overrightarrow{z}_{LU} = \Phi(\overrightarrow{x}_L) - \Phi(\overrightarrow{x}_U)$.

Notice that the only difference with the update in standard SVM is the apparition of y_L, so the dual change becomes

$$\mathcal{D} \quad = \quad \mathcal{D} + \frac{\lambda_L^2}{2}\|\overrightarrow{z}_{LU}\|_2^2 + \lambda_L y_L \langle \overrightarrow{w}, \overrightarrow{z}_{LU}\rangle = \psi(\lambda_L).$$

Differentiating $\psi(\lambda_L)$ and equaling to zero yield the unconstrained optimum

$$\lambda_L \quad = \quad -y_L \frac{\langle \overrightarrow{w}, \overrightarrow{z}_{LU}\rangle}{\|\overrightarrow{z}_{LU}\|_2^2} = -y_L \frac{\Delta_{y_L}}{\|\overrightarrow{z}_{LU}\|_2^2}, \tag{4.30}$$

with $\Delta_{y_L} = \langle \overrightarrow{w}, \overrightarrow{z}_{LU}\rangle$. The same subscript y_L as in GSK is used to emphasize the dependence on the class chosen. Substitution of (4.30) in the dual update gives

$$\mathcal{D} \quad \leftarrow \quad \mathcal{D} - \frac{\Delta_{y_L}^2}{2\|\overrightarrow{z}_{LU}\|_2^2}. \tag{4.31}$$

In order to choose L and U, we should have a descent direction with $\psi'(0) < 0$. As for GSK, this means that $y_L \Delta_{y_L} < 0$, which makes (4.30) positive. To fulfill $\Delta_+ < 0$ and $\Delta_- > 0$, a sensible selection seems to be

$$\begin{aligned} L_+ &= \arg\min_{i\in\mathcal{I}_+}\{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}, \qquad & U_+ &= \arg\max_{i\in\mathcal{I}_+}\{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}, \\ L_- &= \arg\max_{i\in\mathcal{I}_-}\{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}, & U_- &= \arg\min_{i\in\mathcal{I}_-}\{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle\}, \end{aligned}$$

but, as for standard MDM, this has to be refined. By (4.30), with the above choice we obtain $\lambda_L > 0$, so $\lambda_U < 0$, and then it is required that $\alpha_U > 0$, due to the non–negativity constraint of the coefficients. As a result, the refined choice is

Algorithm 5 MDM's algorithm for CH–NPP

while $-y_L \Delta_{y_L} > \epsilon$ **do**
  Choose L_+, L_-, U_+ and U_- with (4.32).
  Select the class for which $-y_L \Delta_{y_L}$ is largest.
  Compute the unconstrained λ_L with (4.30).
  Clip λ if necessary using (4.33).
  Update $\overrightarrow{\alpha}$ with (4.29).
end while

$$\begin{aligned} L_+ &= \arg\min_{i \in \mathcal{I}_+} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \qquad U_+ = \arg\max_{i \in \mathcal{I}_+ : \alpha_i > 0} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \\ L_- &= \arg\max_{i \in \mathcal{I}_-} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \qquad U_- = \arg\min_{i \in \mathcal{I}_- : \alpha_i > 0} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}. \end{aligned} \tag{4.32}$$

Since the expression for $\psi'(\lambda_L) = 0$ is the same as for GSK, the class finally chosen is also the one for which $-y_L \Delta_{y_L}$ is largest. Once it has been chosen, $\lambda_L > 0$, so the only possibility for clipping is because it is too large:

$$\lambda_L \quad \leftarrow \quad \min\{\lambda_L, \alpha_U\}. \tag{4.33}$$

In the optimum it also holds that $\Delta_+^* = 0 = \Delta_-^*$, so that $y_L \Delta_{y_L} = 0$ and no descent direction can be found, exactly as for GSK. This is so because the closest points in two convex hulls are also determined by the points with positive coefficients, which constitute the closest facets of the polytopes. The stopping criterion is hence the same one, as reflected in Algorithm 5.

4.2.3 Extension to RCHs

At first sight, it seems that extending all the above to cover RCH–MNP and RCH–NPP should be very straightforward. In fact, the only difference between CH–MNP and RCH–MNP, and between CH–NPP and RCH–NPP is that the α_i coefficients are bounded above by the value μ. In this section we will deal directly with the NPP formulation, since the MNP one is a particular case of it.

However, this simplicity is only apparent. In the case of the standard GSK algorithm, we optimized on the line segment joining the current estimate $\overrightarrow{w}$ and the point with the smallest margin. This point is always an extreme point of the corresponding convex hull. Then, if we want to generalize this to reduced convex hulls, we need a way to characterize the extreme points of a reduced convex hull, given the factor μ. This is not

trivial, because the shape of the reduced convex hull and the number of extreme points vary considerably depending on this value, as illustrated in Figure 2.4.

Fortunately, we will see in §4.2.3.1 that there is a way to characterize the extreme points of reduced convex hulls. Moreover, it is possible to calculate analytically which is the extreme point with the smallest margin, thus allowing the generalization of GSK into the so–called RCH–GSK method.

As for the MDM algorithm, recall that the basic idea is to choose the point with the smallest margin as in GSK, but also the current support vector with the largest margin. We have said that the point with the smallest margin is also tractable for RCHs, but this "support vector with the largest margin" brings in new difficulties. The problem is that the extreme points of reduced convex hulls turn out to be convex combinations of the extreme points of the standard convex hulls. Hence, there is not a direct correspondence between being a support vector (i.e. having $\alpha_i > 0$) and having an influence in the current estimate inside the reduced convex hull. We will explain this more thoroughly in §4.2.3.2, where a recent attempt to generalize MDM to RCHs in described. This attempt proves unsatisfactory in some cases, getting stuck at suboptimal points. In Chapter 5 we will see how to circumvent the problem outlined previously, arriving to a fully satisfactory RCH–MDM algorithm.

4.2.3.1 RCH–GSK Algorithm

It is clear that the GSK update of $\overrightarrow{w}_+$ ($\overrightarrow{w}_-$) will keep this component of the weight vector in the corresponding reduced convex hull provided that the updating pattern $\Phi(\overrightarrow{x}_L)$ is also in the positive (negative) reduced convex hull.

However, the choices of the indices given in (4.27) make this unlikely. To overcome this, we observe that $\Phi(\overrightarrow{x}_L)$ is in fact a extreme point of the positive (negative) class's convex hull. Thus, in order to generalize GSK, we should have an expression for the extreme points of the positive (negative) class's reduced convex hull. Moreover, this expression should be concrete enough so that we do not have to carry out a combinatorial search over all the points.

The first characterization of the extreme points of reduced convex hull was given by Mavroforakis and Theodoridis in [37], where these authors give a thorough discussion of the geometrical properties of reduced convex hulls. It turns out that these extreme points are of the form

$$\Phi\left(\overrightarrow{\widehat{x}}\right) \;=\; \mu\sum_{k=1}^{K}\Phi\left(\overrightarrow{x}_{p_k}\right)+\sigma\Phi\left(\overrightarrow{x}_{p_{K+1}}\right),$$

where $K=\lfloor 1/\mu\rfloor$, $\sigma=1-K\mu$, $\overrightarrow{p}$ is a permutation of $\mathcal{I}_+$ ($\mathcal{I}_-$), and the hat indicates extreme points in the reduced convex hull. Observe that this formula *per se* is not sufficient, because there is still a combinatorial nature in it, as we can choose any possible permutation. Moreover, this formula is valid just in one sense: every extreme point has this form, but not every point of this form is an extreme point.

Fortunately, it is possible to arrive to an expression for the extreme point with the smallest projection with a greedy approach. Note that the combination above assigns a weight of μ to the first K points of the combination and $1-\mu\lfloor 1/\mu\rfloor$ for the next one (in the particular case where $\lfloor 1/\mu\rfloor = 1/\mu$, this last weight becomes zero, so this point does not influence). Since $\mu > 1-\mu\lfloor 1/\mu\rfloor$, this means that the first K points are more influential than the last one. The weights sum up to 1 in any case, so it is in fact a convex combination.

The greedy approach tells us to assign as much weight as possible to the points with the smallest projections, so that their combination has the smallest possible projection too. This translates into doing the following:

$$\begin{array}{rcl}\Phi\left(\overrightarrow{\widehat{x}}_{L_+}\right) & = & \mu\sum_{k=1}^{K}\Phi\left(\overrightarrow{x}_{p_k^+}\right)+\sigma\Phi\left(\overrightarrow{x}_{p_{K+1}^+}\right)\\ \Phi\left(\overrightarrow{\widehat{x}}_{L_-}\right) & = & \mu\sum_{k=1}^{K}\Phi\left(\overrightarrow{x}_{p_k^-}\right)+\sigma\Phi\left(\overrightarrow{x}_{p_{K+1}^-}\right),\end{array} \tag{4.34}$$

where $\overrightarrow{p}^+$ ($\overrightarrow{p}^-$) is the permutation of $\mathcal{I}_+$ ($\mathcal{I}_-$) that orders the indices by increasing (decreasing) values of $\langle\overrightarrow{w},\Phi(\overrightarrow{x}_i)\rangle$. This is in clear correspondence to (4.27), where for L_+ we looked for the minimum and for L_- we looked for the maximum.

It is worth observing that now there are no explicit indices L_+ and L_-, but in fact we do not need them. All we have to do is to redefine $\overrightarrow{z}_L$ as $\overrightarrow{z}_L=\Phi\left(\overrightarrow{\widehat{x}}_L\right)-\overrightarrow{w}_{y_L}$ once we have chosen the class.

The rest of the algorithm is exactly the same as in GSK, because the same reasonings are valid for choosing the class as the one giving the largest value of $-y_L\Delta_{y_L}$, and for stopping when this quantity is small enough. Thus, it holds that

$$\lambda \;=\; \min\left\{-y_L\frac{\Delta_{y_L}}{\|\overrightarrow{z}_L\|_2^2},1\right\}. \tag{4.35}$$

Algorithm 6 RCH–GSK algorithm for RCH–NPP

while $-y_L\Delta_{y_L} > \epsilon$ **do**
  Calculate $\Phi\left(\overrightarrow{\widetilde{x}}_{L_+}\right)$ and $\Phi\left(\overrightarrow{\widetilde{x}}_{L_-}\right)$ with (4.34).
  Select the class for which $-y_L\Delta_{y_L}$ is largest.
  Calculate λ with (4.35).
  Update $\overrightarrow{\alpha}$ with (4.36).
end while

However, once we have λ, we must note that the update in the coefficients is now different, because $\Phi\left(\overrightarrow{\widetilde{x}}_L\right)$ is a combination of several points:

$$\begin{array}{lcl} \alpha_{p_k} & \leftarrow & (1-\lambda)\alpha_{p_k} + \lambda\mu, \;\; 1 \leq k \leq K, \\ \alpha_{p_{K+1}} & \leftarrow & (1-\lambda)\alpha_{p_{K+1}} + \lambda\sigma, \\ \alpha_i & \leftarrow & (1-\lambda)\alpha_i, \;\; \forall i \in \left\{p_{K+2}, \ldots, p_{|\overrightarrow{p}|}\right\} \\ \alpha_i & \leftarrow & \alpha_i, \;\; \text{otherwise}, \end{array} \tag{4.36}$$

where $|\overrightarrow{p}|$ stands for $|\mathcal{I}_+|$ ($|\mathcal{I}_-|$), depending on the class chosen. The code is summarized in Algorithm 6.

Clearly, RCH–GSK behaves exactly in the same way in the reduced convex hulls than GSK does in the standard convex hulls. There is however a computational overhead in the orderings required for the permutations $\overrightarrow{p}^+$ and $\overrightarrow{p}^-$. Ordering these values has a cost of $\mathcal{O}(N \log N)$, something which GSK is free of. Moreover, in order to be able to perform this orderings as quickly as possible, we need to keep track of the values $\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle = y_i \nabla \mathcal{D}_i$, as for SMO.

Since $\overrightarrow{w}$ is updated in the form $\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda y_L \overrightarrow{z}_L$, the update on $y_i \nabla \mathcal{D}_i$ is

$$y_i \nabla \mathcal{D}_i \;\; \leftarrow \;\; y_i \nabla \mathcal{D}_i + \lambda y_L \langle \overrightarrow{z}_L, \Phi(\overrightarrow{x}_i)\rangle,$$

which has a computational cost of $\mathcal{O}(KN)$ because there are N possible values for i and each one implies K kernel evaluations of the form $\langle \Phi(\overrightarrow{x}_i), \Phi(\overrightarrow{x}_j)\rangle$, due to the decomposition of $\overrightarrow{z}_L$ and (4.34). In contrast, GSK has in this phase a cost of $\mathcal{O}(N)$, as $\Phi(\overrightarrow{x}_L)$ is a single point, not a combination of K (or $K+1$) points. That is, RCH–GSK is K times as expensive as GSK in the updating phase, which can result in a quite slow speed in practice, especially if K is large (i.e., μ is small).

4.2.3.2 An RCH Algorithm Based on MDM

Because of this considerable computational overhead of RCH–GSK when compared to GSK, it would be of interest to obtain an algorithm that had the same cost for both cases, or at least whose cost for the reduced convex hull case compared more favorably to the one for CH–NPP. Although a slight modification of RCH–GSK was proposed in [87] that worked with a single polytope (specifically, the Minkowski difference between the two reduced convex hulls) and that resulted in fewer iterations than RCH–GSK itself, the cost is still too high for practical purposes.

This is the reason why Tao *et al.* [88] tried to generalize the MDM algorithm for RCH–NPP, after they had arrived in [89] to the same generalization for RCH–GSK explained above. Once we know by this last algorithm how to build the point $\Phi\left(\overrightarrow{\widehat{x}}_L\right)$, it is natural to ask if we can do something similar to build a point $\Phi\left(\overrightarrow{\widehat{x}}_U\right)$ that corresponds to MDM's selection (4.32).

In order to proceed analogously, this point $\Phi\left(\overrightarrow{\widehat{x}}_U\right)$ should be a "support vector" (note that we are not dealing with SVMs, but we keep this term to refer to positive coefficients) in the reduced convex hull. This means that it should be a point with a coefficient $\widehat{\alpha}_j > 0$, where this coefficient is also relative to the reduced convex hull, hence the hat and the different index j. However, there is no known translation between the coefficients α_i and these hypothetical coefficients $\widehat{\alpha}_j$.

For instance, imagine that we are optimizing in the positive class, and $\overrightarrow{w}_+$ is equal to an extreme point of the positive class's reduced convex hull. Then we know from the discussion for RCH–GSK that there will be K support vectors with $\alpha_i = \mu > 0$, and possibly another support vector with $\alpha_i = \sigma$. However, in terms of the $\widehat{\alpha}_j$, there would be only one coefficient equal to 1, since we are exactly in that point. The rest would be 0, so the correspondence is seen to be not trivial at all. There is also the additional difficulty of determining which is the size of the hypothetical set of indices $\mathcal{J}$, which should be the number of extreme points of the reduced hull, but this is not known a priori.

After encountering these difficulties, the algorithm in [88] suggests building $\Phi\left(\overrightarrow{\widehat{x}}_L\right)$ as in RCH–GSK applying (4.34), and, instead of trying to build a $\Phi\left(\overrightarrow{\widehat{x}}_U\right)$, choosing U as in MDM, that is, applying (4.32).

Mixing then RCH–GSK and MDM, the update on $\overrightarrow{w}$ is

$$\overrightarrow{w} \leftarrow \overrightarrow{w} + \lambda_L y_L \left(\Phi\left(\overrightarrow{\widehat{x}}_L\right) - \Phi(\overrightarrow{x}_U)\right) = \overrightarrow{w} + \lambda_L y_L \overrightarrow{z}_{LU},$$

Algorithm 7 Tao *et al.*'s algorithm for RCH–NPP

while $-y_L\Delta_{y_L} > \epsilon$ **do**
  Calculate $\Phi\left(\overrightarrow{\widehat{x}}_{L_+}\right)$ and $\Phi\left(\overrightarrow{\widehat{x}}_{L_-}\right)$ with (4.34).
  Choose U_+ and U_- as in MDM, applying (4.32).
  Select the class for which $-y_L\Delta_{y_L}$ is largest.
  Calculate λ with (4.38).
  Update $\overrightarrow{\alpha}$ with (4.37).
end while

where $\overrightarrow{z}_{LU}$ is now defined as $\overrightarrow{z}_{LU} = \Phi\left(\overrightarrow{\widehat{x}}_L\right) - \Phi(\overrightarrow{x}_U)$. Establishing again that $\Delta_{y_L} = \langle\overrightarrow{w}, \overrightarrow{z}_{LU}\rangle$, the class finally chosen is the one for which $-y_L\Delta_{y_L}$ is largest.

This means that the update on $\overrightarrow{\alpha}$ is given by

$$\begin{array}{lcl} \alpha_{p_k} & \leftarrow & \alpha_{p_k} + \lambda\mu, \;\; 1 \leq k \leq K, \\ \alpha_{p_{K+1}} & \leftarrow & \alpha_{p_{K+1}} + \lambda\sigma, \\ \alpha_U & \leftarrow & \alpha_U - \lambda, \\ \alpha_j & \leftarrow & \alpha_j, \;\; \text{otherwise}, \end{array} \tag{4.37}$$

where λ is calculated analogously to MDM as

$$\lambda_L \;\; = \;\; \min\left\{-y_L \frac{\Delta_{y_L}}{\|\overrightarrow{z}_{LU}\|_2^2}, \alpha_U\right\}. \tag{4.38}$$

As with the previous algorithms, the pseudocode is summarized in Algorithm 7.

There are some problems with this algorithm, though. One drawback is that its cost is the same as for RCH–GSK, because the calculation of $\Phi\left(\overrightarrow{\widehat{x}}_L\right)$ requires $\mathcal{O}(N \log N)$ operations for sorting, whereas the update in the gradient implies $\mathcal{O}(KN)$ (more precisely, $(K+1)N$ or even $(K+2)N$) kernel evaluations.

To make things worse, this algorithm is not guaranteed to converge to an optimal solution, not even asymptotically as is the case for RCH–GSK. This problem was not reported in [88], though, but Figure 4.3 makes it clear. Here we depict the same example as in Figure 3.9, so that $\mu = 1/2$. It is assumed that we have already reached the optimum in the negative class, so we can only optimize on the positive one.

The current estimate $\overrightarrow{w}$ is exactly the midpoint between $\Phi(\overrightarrow{x}_4)$ and $\Phi(\overrightarrow{x}_5)$, so that $\alpha_4 = \alpha_5 = 1/2$. The extreme point of the reduced convex hull with smallest projection is $\Phi\left(\overrightarrow{\widehat{x}}_L\right)$, which is the midpoint between $\Phi(\overrightarrow{x}_3)$ and $\Phi(\overrightarrow{x}_5)$. It is clear then than in this situation $U = 4$, as $\Phi(\overrightarrow{x}_4)$ has a larger projection along $\overrightarrow{w}$ than $\Phi(\overrightarrow{x}_5)$.

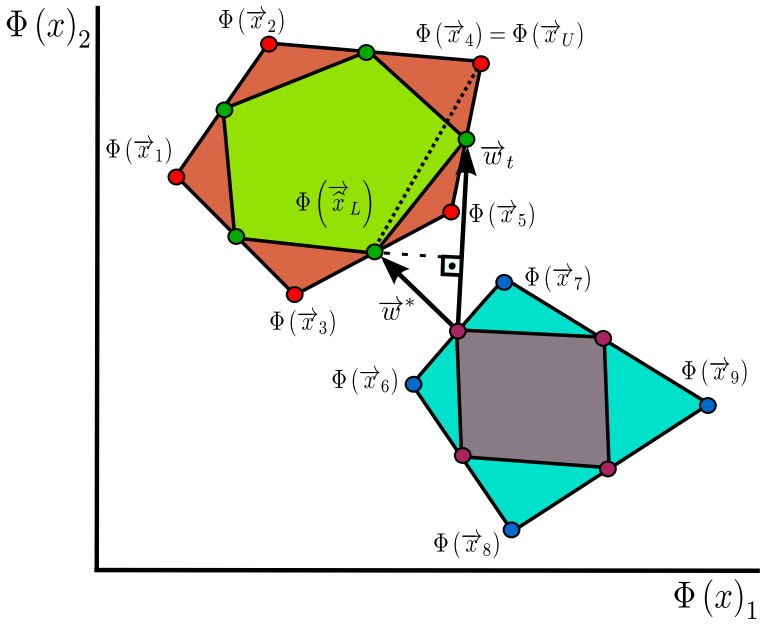

FIGURE 4.3: Illustration of a situation where Algorithm 7 stalls at a suboptimal point, after choosing a non–feasible update direction.

The direction chosen by Algorithm 7 (starting in $\overrightarrow{w}$ and parallel to the one highlighted with a dashed line) is not feasible, because all its points are outside the reduced convex hull, as we are already on its border. Thus, the algorithm stalls at a suboptimal $\overrightarrow{w}$, which is very different from the optimal $\overrightarrow{w}^*$.

In the next chapter we will propose an adaptation of the MDM algorithm for RCH–NPP that, building on the same clipping strategy than SMO, does not suffer from this problem of stalling at suboptimal points. Moreover, the complexity of the algorithm is maintained in $\mathcal{O}(2N)$, as for standard MDM.

Chapter 5

Unification of SVM and Geometric Formulations

All the previous chapters have provided us with the necessary knowledge to understand the two main contributions of this dissertation, which are: 1) the unification of all the SVM and geometric formulations of Chapter 3 into a common problem, and 2) showing the convergence of an SMO algorithm adapted for this common problem. The present chapter deals with the first one, while the proof of convergence is the subject of Chapter 6.

Before arriving to this general formulation, we will analyze the inherent relationship between solving a nearest point problem and finding a maximal margin hyperplane. In order to understand this, we will proceed in a sequential way: first, we will see how we can arrive to a generalization of the MDM algorithm (§4.2.2.2) for the RCH–NPP problem, which we will term RCH–MDM. It was explained in §4.2.3.2 why the proposal in [88] is not satisfactory. After this, we will explain why NPP and SVC can be considered to be equivalent problems. This is easily inferred by means of the KKT optimality conditions for the different problems. Then, we will point out some consequences, the most important one being that the suggested RCH–MDM is a particular version of the SMO algorithm described in §4.1.1.1.

Having completed this section, we will derive both a primal and a dual formulation that include as particular cases all the primal and dual formulations enumerated in Chapter 3. The resulting primal problem is also intractable when using a non–linear kernel, so it is the general dual that has to be solved. Not surprisingly, adapting the SMO algorithm to this dual results in a method that encompasses all the particular SMO versions tailored to the specific problems included in this general formulation.

5.1 The Geometry behind SVMs

The publication of the works by K.P. Bennett and E.J. Bredensteiner [24, 90] was quite celebrated, for it showed that Support Vector Classification and solving nearest point problems were very intimately related tasks. This opened a new line of research whose main objective was to train SVMs based on geometric ideas. As a result, the Gilbert and MDM algorithms were soon "rescued" and applied for SVC in the papers [27, 86], as explained in the previous chapter.

However, as years passed this line of research lost popularity. It regained it with the suggestion of algorithms to deal with reduced convex hulls [37, 88, 89], but, since then, not too much attention has been given to geometry. Even so, we explored these topics in [58]. The present section can be considered as a natural extension of this master's thesis.

5.1.1 A Proper RCH–MDM Algorithm

In §4.2.3.2 we analyzed Algorithm 7 and realized that this method has two important drawbacks, namely: 1) it is not guaranteed to arrive to an optimum, 2) its computational cost per iteration is $\mathcal{O}((K + \log N)N)$, which is rather expensive. Although the geometrical framework provided in [37] for reduced convex hulls apparently means that the cost $\mathcal{O}(KN)$ is insurmountable, because every extreme point is a combination of K points, we are able to circumvent this difficulty.

The basic idea behind this algorithm can be summarized with the following thought: "if geometry does not work, why not trying a clipping strategy like the one followed by SMO?". In fact, SMO clips adequately the updates on $\overrightarrow{\alpha}$, so that the bounds are met. It is thus natural to consider whether simply clipping the GSK and MDM updates would result in effective algorithms for RCH–NPP.

This idea was suggested initially in [91], and more deeply analyzed in [92]. It was observed that this strategy is not valid for GSK, so that modification is ignored in the sequel. Fortunately, it is indeed valid for MDM. Even if it may be argued that the geometrical component of the algorithm is lost, the benefits from this approach far outweigh the loss of visual interpretation.

Recalling the description of MDM for CH–NPP in §4.2.2.2, we had to refine the selection of U_+ and U_- in (4.32) so as to ensure that the coefficient α_U remains non–negative. There was no concern about L_+ (L_-), which could be selected as the global minimum

Algorithm 8 MDM's algorithm for RCH–NPP

while $-y_L\Delta_{y_L} > \epsilon$ **do**
  Choose L_+, L_-, U_+ and U_- with (5.1).
  Select the class for which $-y_L\Delta_{y_L}$ is largest.
  Compute the unconstrained λ_L as in MDM, with (4.30).
  Clip λ if necessary using (5.2).
  Update $\overrightarrow{\alpha}$ as in MDM, with (4.29).
end while

(maximum) of $\langle \overrightarrow{w}, \Phi(\overrightarrow{x}) \rangle$ over $\mathcal{I}_+$ ($\mathcal{I}_-$), because the constraint $\sum_{i=1}^{N} \alpha_i = 1$ implicitly ensures that α_L cannot grow arbitrarily.

Nevertheless, in RCH–NPP this is different, because this constraint does not ensure that α_L remains less than or equal to μ (provided that $\mu < 1$, which is the usual case). Since the selection guarantees that $\lambda_L > 0$, if we are to select a single point as $\Phi(\overrightarrow{x}_L)$, we need to change (4.32) to

$$\begin{aligned} L_+ &= \arg\min_{i \in \mathcal{I}_+ : \alpha_i < \mu} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \qquad & U_+ &= \arg\max_{i \in \mathcal{I}_+ : \alpha_i > 0} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \\ L_- &= \arg\max_{i \in \mathcal{I}_- : \alpha_i < \mu} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}, \qquad & U_- &= \arg\min_{i \in \mathcal{I}_- : \alpha_i > 0} \{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle\}. \end{aligned} \tag{5.1}$$

In addition to this, there is a further possibility of clipping if λ_L turns out to be greater than $\mu - \alpha_L$, so (4.33) is changed to

$$\lambda_L \leftarrow \min\{\lambda_L, \alpha_U, \mu - \alpha_L\}. \tag{5.2}$$

The stopping criterion is still valid, for reasons that will become clear in §5.1.2. The resulting method is shown in Algorithm 8. We will refer to this algorithm in the sequel as RCH–MDM.

These two simple enhancements over the MDM algorithm for CH–NPP allow it to perform well for RCH–NPP. In fact, since now we are selecting L and U as points from the original sample, the number of kernel operations per iteration is still $2N$, as for standard MDM. This is not the case for RCH–GSK, whose iterations carry out KN kernel iterations, K times more than CH–GSK.

It might seem that this improvement on complexity should come at a cost of iterations, and that RCH–GSK should need less iterations to converge than RCH–MDM. As a matter of fact, RCH–GSK retained the geometrical justification of GSK, whereas RCH–MDM has lost it. We are not choosing the extreme points $\Phi\left(\overrightarrow{\widetilde{x}}_L\right)$ nor $\Phi\left(\overrightarrow{\widetilde{x}}_U\right)$, but

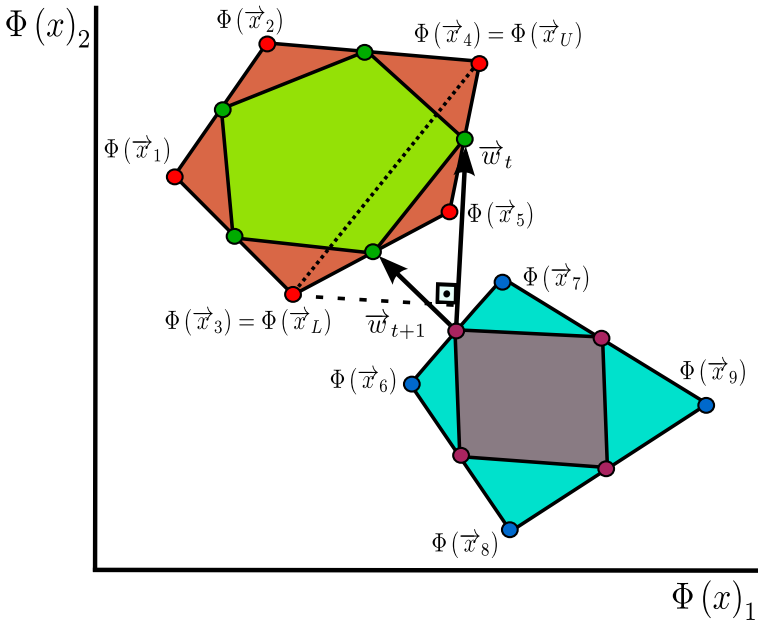

FIGURE 5.1: Illustration of the same situation of Figure 4.3. Whereas Algorithm 7 could not progress to the optimal solution from the position shown, Algorithm 8 arrives in a single iteration.

considering instead the extreme points from the original training set. Thus, RCH–MDM is not the direct translation of MDM to reduced convex hulls, while RCH–GSK is indeed the translation of GSK.

We will see in the next section, after having established the relationship among NPP, SVC and ν–SVC, that RCH–MDM has a better performance than RCH–GSK, both in terms of number of iterations and kernel operations. It is necessary to clarify this relationship first, so as to test the algorithms in equal conditions.

For the time being, let us examine again the example of Figure 4.3, and illustrate that Algorithm 8 does not stall in this case. Recall that the current estimate was the midpoint between $\Phi(\overrightarrow{x}_4)$ and $\Phi(\overrightarrow{x}_5)$, so that $\alpha_4 = \alpha_5 = 1/2$.

While the selection for U is the same as in Algorithm 7, that is, $U = 4$, the key is that now we simply pick $L = 3$, as it is the point of the original convex hull (not the reduced one) with smallest projection. This makes the direction of RCH–MDM (again shown with a dashed line) feasible, as it is now parallel to the edge of the reduced convex hull we are moving along. In fact, we arrive in a single step to the optimal $\overrightarrow{w}^*$, while Tao *et al.*'s method got stuck in the previous estimate. This is illustrated in Figure 5.1.

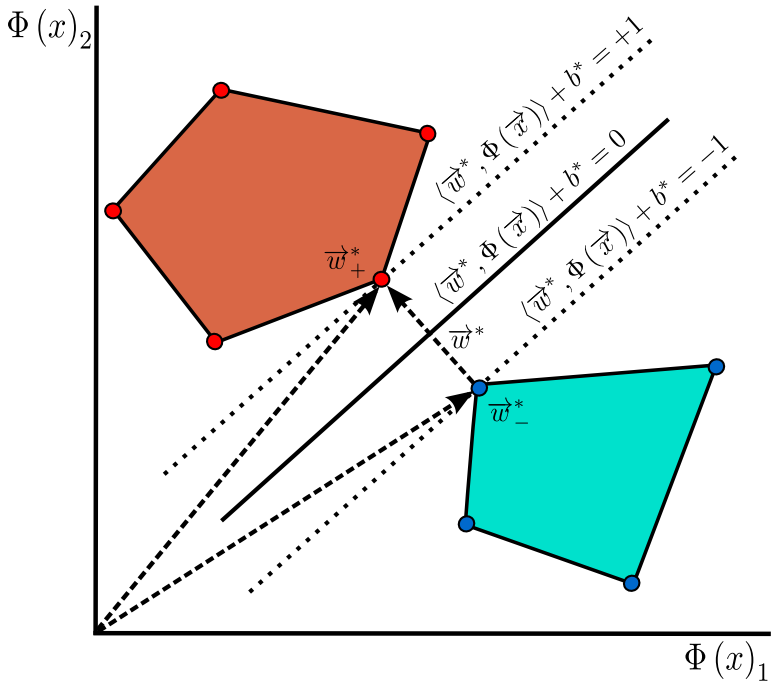

FIGURE 5.2: Relationship between CH–NPP and hard–margin SVC. The maximal margin hyperplane bisects the segment joining the two closest points in the convex hulls of each class.

5.1.2 The Relationship among NPP, SVC and ν–SVC

Not surprisingly, these three problems are closely related. Basically speaking, ν–SVC (Section 3.1.4) is like SVC (§3.1.3 and §3.1.2), but allowing for a margin ρ that can be different from 1. It also replaces the parameter C with a new parameter ν. As for NPP, it is visually intuitive that finding the closest points in two convex hulls is similar to finding the maximal margin hyperplane.

This idea is illustrated in Figure 5.2 for the hard–margin case. Here the classes are linearly separable, so the convex hulls formed by their respective points do not overlap. It becomes visually evident that the optimal hyperplane has as its normal the vector joining the closest points from each hull, that is, $\vec{w}^* = \vec{w}^*_+ - \vec{w}^*_-$. Regarding the bias, this hyperplane should bisect this vector $\vec{w}^*$, so using geometry

$$\left\langle \vec{w}^*, \frac{\vec{w}^*_+ + \vec{w}^*_-}{2} \right\rangle + b^* = 0 \quad \Rightarrow \quad b^* = \frac{\left\|\vec{w}^*_-\right\|_2^2 - \left\|\vec{w}^*_+\right\|_2^2}{2}.$$

Let us further characterize this relationship. We will follow the lines of [24], starting with the l_2 case, which is a bit easier to analyze than the l_1 one.

5.1.2.1 The Relationship among CH–NPP, l_2–SVC and l_2–ν–SVC

It is useful to formulate the following additional problem:

$$\begin{aligned} \min_{\overrightarrow{w},\gamma,\eta,\overrightarrow{\xi}} \quad & \frac{1}{2}\left\|\overrightarrow{w}\right\|_2^2 - \gamma + \eta + \frac{\mu}{2}\sum_{i=1}^{N}\xi_i^2 \\ \text{s.t.} \quad & \begin{cases} \left\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\right\rangle \geq \gamma - \xi_i, & \forall i \in \mathcal{I}_+, \\ \left\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\right\rangle \leq \eta + \xi_i, & \forall i \in \mathcal{I}_-. \end{cases} \end{aligned} \tag{5.3}$$

This problem tries to find two parallel "support" hyperplanes $\left\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}\right)\right\rangle = \gamma$ and $\left\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}\right)\right\rangle = \eta$ that leave the positive and the negative classes in their positive and negative halfspaces, respectively. Additionally, we allow for errors with the slacks ξ_i, which are weighted with the trade–off parameter μ. We do so because the objective function is a compromise between two conflicting goals: keeping these slacks at bay and trying to separate the hyperplanes as much as possible, as the distance between them is $(\gamma - \eta)/\left\|\overrightarrow{w}\right\|_2$. The value of μ quantifies how much importance we give to hyperplane separation and how much to error limitation.

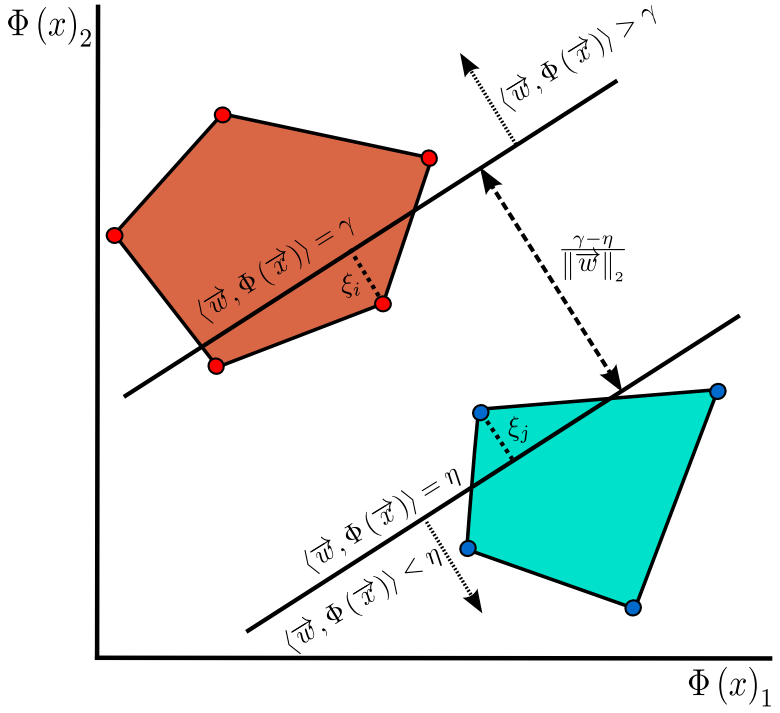

FIGURE 5.3: Illustration of the CH–Margin problem. Two points that lie in the wrong halfspaces of the hyperplanes are highlighted with the slack variables associated to them.

In the sequel, we will refer to problem (5.3) as CH–Margin, which is illustrated in Figure 5.3. Dualizing this problem, it turns out that we arrive exactly to the CH–NPP formulation (3.70), with the modified kernel matrix $\widetilde{K}_{ij} = K_{ij} + \delta_{ij}/\mu$. In order to get CH–NPP with the standard kernel matrix, its primal is the problem above with no slack variables, that is, its hard–margin version:

$$\begin{aligned} \min_{\overrightarrow{w},\gamma,\eta} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \gamma + \eta \\ \text{s.t.} \quad & \begin{cases} \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle \geq \gamma, & \forall i \in \mathcal{I}_+, \\ \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle \leq \eta, & \forall i \in \mathcal{I}_-. \end{cases} \end{aligned} \tag{5.4}$$

Thus, we have to show the relationship joining CH–Margin, l_2–ν–SVC and l_2–SVC (the hard–margin versions are omitted in the sequel, as we will assume linearly inseparable sets). The easiest way to do this is comparing the KKT optimality conditions for each problem.

Let us start with l_2–SVC. Recalling (3.19), the KKT conditions are:

$$\begin{aligned} \text{S} \quad & \begin{cases} \overrightarrow{w}^* = \sum_{i=1}^N \alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^N \alpha_i^* y_i = 0, \end{cases} \\ \text{DF} \quad & \left\{ \alpha_i^* \geq 0, \quad i = 1, \ldots, N, \right. \\ \text{PF} \quad & \left\{ y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) \geq 1 - \frac{\alpha_i^*}{C}, \quad i = 1, \ldots, N, \right. \\ \text{CS} \quad & \left\{ \alpha_i^* \left[y_i(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*) - 1 + \frac{\alpha_i^*}{C} \right] = 0, \quad i = 1, \ldots, N, \right. \end{aligned} \tag{5.5}$$

where S stands for stationarity, DF for dual feasibility, PF for primal feasibility and CS for complementary slackness. Here we use the star symbol to denote a primal optimal solution $\left(\overrightarrow{w}^*, b^*, \overrightarrow{\xi}^*\right)$ and a dual optimal solution $\overrightarrow{\alpha}^*$. Since we are using the l_2 penalty, we have $\overrightarrow{\xi}^* = \overrightarrow{\alpha}^*/C$ by virtue of (3.13).

Moving to l_2–ν–SVC, the KKT conditions were not treated in §3.1.4, but it is easy to see that they are

$$
\begin{array}{ll}
\text{S} & \begin{cases} \overrightarrow{w}' = \sum_{i=1}^{N} \alpha_i' y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^{N} \alpha_i' y_i = 0, \\ \sum_{i=1}^{N} \alpha_i' = \nu, \end{cases} \\
\text{DF} & \begin{cases} \alpha_i' \geq 0, \quad i = 1, \ldots, N, \end{cases} \\
\text{PF} & \begin{cases} y_i \left(\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right) \rangle + b' \right) \geq \rho' - N\alpha_i', \quad i = 1, \ldots, N, \end{cases} \\
\text{CS} & \begin{cases} \alpha_i' \left[y_i (\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right) \rangle + b') - \rho' + N\alpha_i' \right] = 0, \quad i = 1, \ldots, N, \end{cases}
\end{array}
\tag{5.6}
$$

where now we use the prime symbol to denote a primal optimal solution $\left(\overrightarrow{w}', b', \rho', \overrightarrow{\xi}'\right)$ and a dual optimal solution $\overrightarrow{\alpha}'$. In this case we get $\overrightarrow{\xi}' = N\overrightarrow{\alpha}'$ by (3.35).

Finally, the KKT conditions for the CH–Margin problem have the form

$$
\begin{array}{ll}
\text{S} & \begin{cases} \overrightarrow{w}^o = \sum_{i=1}^{N} \alpha_i^o y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{\mathcal{I}_+} \alpha_i^o = \sum_{\mathcal{I}_-} \alpha_i^o = 1, \end{cases} \\
\text{DF} & \begin{cases} \alpha_i^o \geq 0, \quad i = 1, \ldots, N, \end{cases} \\
\text{PF} & \begin{cases} \langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right) \rangle \geq \gamma^o - \frac{\alpha_i^o}{\mu}, \quad \forall i \in \mathcal{I}_+, \\ \langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right) \rangle \leq \eta^o + \frac{\alpha_i^o}{\mu}, \quad \forall i \in \mathcal{I}_-, \end{cases} \\
\text{CS} & \begin{cases} \alpha_i^o \left[\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right) \rangle - \gamma^o + \frac{\alpha_i^o}{\mu} \right] = 0, \quad \forall i \in \mathcal{I}_+, \\ \alpha_i^o \left[\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right) \rangle - \eta^o - \frac{\alpha_i^o}{\mu} \right] = 0, \quad \forall i \in \mathcal{I}_-, \end{cases}
\end{array}
\tag{5.7}
$$

where it is now an o superscript which indicates a primal optimal solution $\left(\overrightarrow{w}^o, \gamma^o, \eta^o, \overrightarrow{\xi}^o\right)$ and a dual optimal solution $\overrightarrow{\alpha}^o$, with $\overrightarrow{\xi}^o = \overrightarrow{\alpha}^o/\mu$ as a result of equalling to zero the gradient of the Lagrangian of (5.3).

We will show now the relationship among these problems with three propositions. This is in the spirit of [24], where the authors showed the relationship between CH–Margin and l_2–SVC. However, we complete the picture with the inclusion of l_2–ν–SVC.

Proposition 5.1. *The KKT conditions of l_2–ν–SVC become the KKT conditions of CH–Margin by defining $\lambda = 2/\nu = 2/\sum_{i=1}^{N} \alpha_i'$ and making the substitutions $\overrightarrow{\alpha}^o = \lambda\overrightarrow{\alpha}'$, $\overrightarrow{w}^o = \lambda\overrightarrow{w}'$, $\gamma^o = \lambda(\rho' - b')$, $\eta^o = \lambda(-\rho' - b')$, $\mu = 1/N$.*

Proof. Multiplying (5.6) by λ except for the CS conditions, which are multiplied by λ^2, we get

$$
\begin{aligned}
&\text{S} && \begin{cases} \lambda \overrightarrow{w}' = \sum_{i=1}^{N} \lambda \alpha_i' y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^{N} \lambda \alpha_i' y_i = 0, \\ \sum_{i=1}^{N} \lambda \alpha_i' = \lambda \nu, \end{cases} \\
&\text{DF} && \left\{ \lambda \alpha_i' \geq 0, \quad i = 1, \ldots, N, \right. \\
&\text{PF} && \left\{ y_i \lambda \left(\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right) \rangle + b' \right) \geq \lambda(\rho' - N\alpha_i'), \quad i = 1, \ldots, N, \right. \\
&\text{CS} && \left\{ \lambda \alpha_i' \left[y_i \lambda (\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right) \rangle + b') - \lambda(\rho' + N\alpha_i') \right] = 0, \quad i = 1, \ldots, N. \right.
\end{aligned}
$$

which, by the substitutions in the proposition, yields

$$
\begin{aligned}
&\text{S} && \begin{cases} \overrightarrow{w}^o = \sum_{i=1}^{N} \alpha_i^o y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^{N} \alpha_i^o y_i = 0, \\ \sum_{i=1}^{N} \alpha_i^o = 2, \end{cases} \\
&\text{DF} && \left\{ \alpha_i^o \geq 0, \quad i = 1, \ldots, N, \right. \\
&\text{PF} && \left\{ y_i \left(\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right) \rangle + \lambda b' \right) \geq \lambda \rho' - N\alpha_i^o, \quad i = 1, \ldots, N, \right. \\
&\text{CS} && \left\{ \alpha_i^o \left[y_i \left(\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right) \rangle + \lambda b' \right) - \lambda \rho' + N\alpha_i^o \right] = 0, \quad i = 1, \ldots, N, \right.
\end{aligned}
$$

Mixing the two constraints $\sum_{i=1}^{N} \alpha_i^o y_i = 0$ and $\sum_{i=1}^{N} \alpha_i^o = 2$ can be written as $\sum_{\mathcal{I}_+} \alpha_i^o = \sum_{\mathcal{I}_-} \alpha_i^o = 1$, so we already have the S and DF conditions of (5.7). Finally, splitting the PF and CS conditions in the different cases $y_i = +1 \; \forall i \in \mathcal{I}_+$ and $y_i = -1 \; \forall i \in \mathcal{I}_-$ gives the PF and CS conditions of (5.7), and we are done. □

This means that any optimal solution for the l_2–ν–SVC problem (3.33), rescaled by the factor $\lambda = 2/\nu$, is also an optimal solution for the CH–Margin problem (5.3) with $\mu = 1/N$.

We can also prove a similar proposition relating l_2–ν–SVC and l_2–SVC:

Proposition 5.2. *The KKT conditions of l_2–ν–SVC become the KKT conditions of l_2–SVC by defining $\lambda = 1/\rho' = \sum_{i=1}^{N} \alpha_i'/(\rho'\nu)$ and making the substitutions $\overrightarrow{\alpha}^* = \lambda \overrightarrow{\alpha}'$, $\overrightarrow{w}^* = \lambda \overrightarrow{w}'$, $b^* = \lambda b'$, $\xi^* = \lambda \xi'$, $C = 1/N$.*

Proof. Multiplying the S, DF and PF conditions of (5.6) by λ and the CS conditions by λ^2 yields the same as before:

$$
\begin{aligned}
&\text{S} && \begin{cases} \lambda \overrightarrow{w}' = \sum_{i=1}^{N} \lambda \alpha_i' y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^{N} \lambda \alpha_i' y_i = 0, \\ \sum_{i=1}^{N} \lambda \alpha_i' = \lambda \nu, \end{cases} \\
&\text{DF} && \left\{ \lambda \alpha_i' \geq 0, \quad i = 1, \ldots, N, \right. \\
&\text{PF} && \left\{ y_i \lambda \left(\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right) \rangle + b' \right) \geq \lambda(\rho' - N\alpha_i'), \quad i = 1, \ldots, N, \right. \\
&\text{CS} && \left\{ \lambda \alpha_i' \left[y_i \lambda (\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right) \rangle + b') - \lambda(\rho' + N\alpha_i') \right] = 0, \quad i = 1, \ldots, N. \right.
\end{aligned}
$$

which, by the substitutions in the proposition, yields

$$
\begin{aligned}
&\text{S} && \begin{cases} \overrightarrow{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^{N} \alpha_i^* y_i = 0, \\ \frac{\sum_{i=1}^{N} \alpha_i'}{\rho'} = \frac{\sum_{i=1}^{N} \alpha_i'}{\rho'}, \end{cases} \\
&\text{DF} && \left\{ \alpha_i^* \geq 0, \quad i = 1, \ldots, N, \right. \\
&\text{PF} && \left\{ y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + \lambda b^* \right) \geq 1 - \frac{\alpha_i^*}{C}, \quad i = 1, \ldots, N, \right. \\
&\text{CS} && \left\{ \alpha_i^* \left[y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + \lambda b^* \right) - 1 + \frac{\alpha_i^*}{C} \right] = 0, \quad i = 1, \ldots, N, \right.
\end{aligned}
$$

which is obviously the same as (5.5), after removal of the trivial constraint $\sum_{i=1}^{N} \alpha_i'/\rho' = \sum_{i=1}^{N} \alpha_i'/\rho'$. □

This means that any optimal solution for the l_2–ν–SVC problem (3.33), rescaled by the factor $\lambda = 1/\rho'$, is also an optimal solution for the l_2–SVC problem (5.3) with $C = 1/N$.

Combining Propositions 5.1 and 5.2 gives

Corollary 5.3 ([24], Theorem 4.2)**.** *The KKT conditions of CH–Margin become the KKT conditions of l_2–SVC by defining $\lambda = 2/(\gamma^o - \eta^o) = \sum_{\mathcal{I}_+} \alpha_i^* = \sum_{\mathcal{I}_-} \alpha_i^*$ and making the substitutions $\overrightarrow{\alpha}^* = \lambda \overrightarrow{\alpha}^o$, $\overrightarrow{w}^* = \lambda \overrightarrow{w}^o$, $b^* = -\lambda(\gamma^o + \eta^o)/2$, $\xi^* = \lambda \xi^o$, $C = \mu$.*

The relationship is summarized in Figure 5.4. When a change of variables is required, it is indicated explicitly with the superscript of the problem at the beginning of the arrow. For example, from b', ρ' we obtain γ', η', which become γ^o, η^o when multiplied by $\lambda = 2/\nu$. Observe that, in a sense, l_2–ν–SVC looks like a more general formulation: by rescaling it with quantities only dependent on itself, we can arrive to the other two formulations. Nevertheless, the reverse is not possible; we cannot go from CH–Margin to l_2–ν–SVC via a scale factor only dependent on the solution of CH–Margin. We will resume this topic in a while, when the relationship for the l_1 case is clarified.

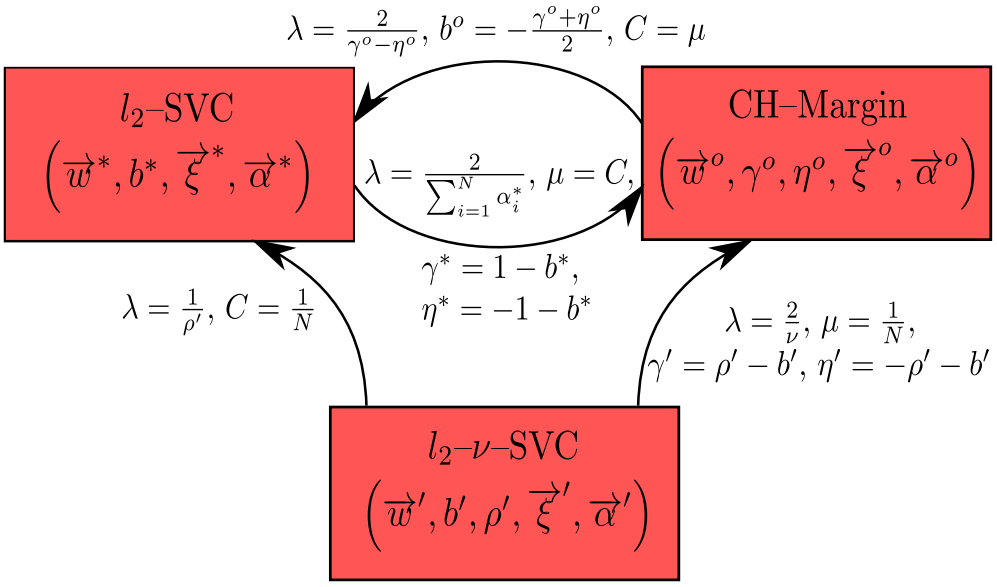

FIGURE 5.4: Relationship between the l_2–SVC, l_2–ν–SVC and CH–Margin optimal solutions. The factor λ indicates the scale factor in the direction of its respective arrow.

Before moving to that case, let us briefly analyse again the KKT conditions of l_2–SVC, l_2–ν–SVC and CH–Margin. Starting with l_2–SVC, assume that we are solving the dual formulation in $\overrightarrow{\alpha}$, so that we automatically fulfil the S and DF conditions. According to the discussion in §3.1.2, we can summarize the PF and CS ones as follows:

$$\begin{aligned} y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &\geq 1 - \tfrac{\alpha_i^*}{C} \quad \forall i : \alpha_i^* = 0, \\ y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &= 1 - \tfrac{\alpha_i^*}{C} \quad \forall i : \alpha_i^* > 0. \end{aligned}$$

Since it must be the case that either $\alpha_i^* = 0$ or $\alpha_i^* > 0$, we can rewrite the above as

$$\begin{aligned} y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &\geq 1 - \tfrac{\alpha_i^*}{C} \quad \forall i, \\ y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &= 1 - \tfrac{\alpha_i^*}{C} \quad \forall i : \alpha_i^* > 0. \end{aligned}$$

The gradient of the dual is given by

$$\nabla \mathcal{D} \;=\; \widetilde{\Omega}\overrightarrow{\alpha}, \tag{5.8}$$

where $\widetilde{\Omega}$ is the matrix with elements $\widetilde{\Omega}_{ij} = y_i y_j (K_{ij} + \delta_{ij}/C)$, according to (3.15). Then the i–th element of this gradient is given by

$$\nabla\mathcal{D}_i \quad = \quad \left(\widetilde{\Omega}\,\overrightarrow{\alpha}\right)_i = y_i \sum_{j=1}^{N} \alpha_j y_j \left(K_{ij} + \frac{\delta_{ij}}{C}\right) = y_i \langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle + \frac{\alpha_i}{C}. \tag{5.9}$$

It is easy to see now that, separating for the cases where $y_i = \pm 1$, we get

$$\begin{aligned} y_i \nabla\mathcal{D}_i^* &\geq -b^* \quad \forall i \in \mathcal{I}_+, \\ y_i \nabla\mathcal{D}_i^* &\leq -b^* \quad \forall i \in \mathcal{I}_-, \\ y_i \nabla\mathcal{D}_i^* &= -b^* \quad \forall i \in \mathcal{I}_+ : \alpha_i^* > 0, \\ y_i \nabla\mathcal{D}_i^* &= -b^* \quad \forall i \in \mathcal{I}_- : \alpha_i^* > 0, \end{aligned}$$

where $\nabla\mathcal{D}^*$ is the shorthand for $\nabla\mathcal{D}(\overrightarrow{\alpha}^*)$. Combining the above means that

$$\min_{i:y_i=+1\vee(y_i=-1\wedge\alpha_i^*>0)} \{y_i \nabla\mathcal{D}_i^*\} \geq -b^* \geq \max_{i:(y_i=+1\wedge\alpha_i^*>0)\vee y_i=-1} \{\nabla y_i \mathcal{D}_i^*\}.$$

In the fashion of what was made for SMO in §4.1.1.1, let us define the sets

$$\begin{aligned} \mathcal{I}_{L^*} &= \{i : y_i = +1 \vee (y_i = -1 \wedge \alpha_i^* > 0)\}, \\ \mathcal{I}_{U^*} &= \{i : (y_i = +1 \wedge \alpha_i^* > 0) \vee y_i = -1\}, \end{aligned} \tag{5.10}$$

as well as the indices

$$L^* = \arg\min_{i\in\mathcal{I}_{L^*}} \{y_i \nabla\mathcal{D}_i^*\}, \qquad U^* = \arg\max_{i\in\mathcal{I}_{U^*}} \{y_i \nabla\mathcal{D}_i^*\}, \tag{5.11}$$

and the value $\Delta^* = y_{U^*}\nabla\mathcal{D}_{U^*} - y_{L^*}\nabla\mathcal{D}_{L^*} = y_U \left\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}(\overrightarrow{x}_{U^*})\right\rangle - y_L \left\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}(\overrightarrow{x}_{L^*})\right\rangle$ in view of (3.16) and (3.17). Then we can finally write the KKT conditions in the single equation

$$\max_{\mathcal{I}_{U^*}} \{y_i \nabla\mathcal{D}_i^*\} \leq \min_{\mathcal{I}_{L^*}} \{y_i \nabla\mathcal{D}_i^*\} \quad \Rightarrow \quad \Delta^* \leq 0. \tag{5.12}$$

The case of CH–Margin is also similar, but it is worth detailing it. Assuming again that we are solving its dual (i.e. CH–NPP), the S and DF conditions of (5.7) are automatically fulfilled, so we are left with the PF and CS ones. Combining them, we get

$$\begin{aligned}
&\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle \geq \gamma^o - \tfrac{\alpha_i^o}{\mu}, \quad \forall i \in \mathcal{I}_+,\\
&\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle = \gamma^o - \tfrac{\alpha_i^o}{\mu}, \quad \forall i \in \mathcal{I}_+ : \alpha_i^o > 0,\\
&\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle \leq \eta^o + \tfrac{\alpha_i^o}{\mu}, \quad \forall i \in \mathcal{I}_-,\\
&\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle = \eta^o + \tfrac{\alpha_i^o}{\mu}, \quad \forall i \in \mathcal{I}_- : \alpha_i^o > 0.
\end{aligned}$$

In this case the gradient is given by

$$\nabla \mathcal{D}_i \;=\; \left(\widetilde{\Omega}\,\overrightarrow{\alpha}\right)_i = y_i \sum_{j=1}^{N} \alpha_j y_j \left(K_{ij} + \frac{\delta_{ij}}{\mu}\right) = y_i \langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle + \frac{\alpha_i}{\mu}, \tag{5.13}$$

so that we can rewrite the formulas above as

$$\begin{aligned}
&y_i \nabla \mathcal{D}_i^o \geq \gamma^o, \quad && \forall i \in \mathcal{I}_+,\\
&y_i \nabla \mathcal{D}_i^o = \gamma^o, \quad && \forall i \in \mathcal{I}_+ : \alpha_i^o > 0,\\
&y_i \nabla \mathcal{D}_i^o \geq -\eta^o, \quad && \forall i \in \mathcal{I}_-,\\
&y_i \nabla \mathcal{D}_i^o = -\eta^o, \quad && \forall i \in \mathcal{I}_- : \alpha_i^o > 0.
\end{aligned}$$

Defining the sets

$$\begin{aligned}
\mathcal{I}_{L_+^o} &= \{i : y_i = +1\}, \qquad \mathcal{I}_{U_+^o} = \{i : (y_i = +1 \wedge \alpha_i^o > 0)\},\\
\mathcal{I}_{L_-^o} &= \{i : y_i = -1\}, \qquad \mathcal{I}_{U_-^o} = \{i : (y_i = -1 \wedge \alpha_i^o > 0)\},
\end{aligned} \tag{5.14}$$

together with the indices

$$L_+^o = \underset{i \in \mathcal{I}_{L_+^o}}{\arg\min}\{y_i \nabla \mathcal{D}_i^o\}, \qquad U_+^o = \underset{i \in \mathcal{I}_{U_+^o}}{\arg\max}\{y_i \nabla \mathcal{D}_i^o\}, \tag{5.15}$$

$$L_-^o = \underset{i \in \mathcal{I}_{L_-^o}}{\arg\min}\{y_i \nabla \mathcal{D}_i^o\}, \qquad U_-^o = \underset{i \in \mathcal{I}_{U_-^o}}{\arg\max}\{y_i \nabla \mathcal{D}_i^o\}, \tag{5.16}$$

and the values $\Delta_+^o = y_{U_+^o} \nabla \mathcal{D}_{U_+^o} - y_{L_+^o} \nabla \mathcal{D}_{L_+^o}$ and $\Delta_-^o = y_{U_-^o} \nabla \mathcal{D}_{U_-^o} - y_{L_-^o} \nabla \mathcal{D}_{L_-^o}$, we can derive

$$\begin{aligned}
\max_{\mathcal{I}_{U_+^o}} \{y_i \nabla \mathcal{D}_i^o\} \leq \min_{\mathcal{I}_{L_+^o}} \{y_i \nabla \mathcal{D}_i^o\} \quad &\Rightarrow \quad \Delta_+^o \leq 0,\\
\max_{\mathcal{I}_{L_-^o}} \{y_i \nabla \mathcal{D}_i^o\} \leq \min_{\mathcal{I}_{U_-^o}} \{y_i \nabla \mathcal{D}_i^o\} \quad &\Rightarrow \quad \Delta_-^o \leq 0,
\end{aligned}$$

so that in the end the KKT conditions of CH–Margin can be summarized in

$$\Delta^o = \max\{\Delta^o_+, \Delta^o_-\} \leq 0. \tag{5.17}$$

Operating in the same way for l_2–ν–SVC we can arrive to

$$\Delta' = \max\{\Delta'_+, \Delta'_-\} \leq 0, \tag{5.18}$$

where now $\widetilde{\Omega}_{ij} = y_i y_j (K_{ij} + N\delta_{ij})$, by virtue of the dual (3.36).

Now that we have clarified the relationship of the l_2 problems, as well as their KKT conditions, let us move to the l_1 formulations.

5.1.2.2 The Relationship among RCH–NPP, l_1–SVC and l_1–ν–SVC

With a very similar scheme to the one used for the l_2 case, it turns out that RCH–NPP, l_1–SVC and l_1–ν–SVC can be considered to be equivalent problems. Although it is not so clear visually that l_1–SVC can be formulated as an NPP as in Figure 5.2, this is indeed the case, this time working with the reduced convex hulls of the samples.

To use it as an intermediary, let us formulate an analogous problem to (5.3). This is

$$\begin{aligned} \min_{\overrightarrow{w},\gamma,\eta,\overrightarrow{\xi}} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 - \gamma + \eta + \mu \sum_{i=1}^{N} \xi_i \\ s.t. \quad & \begin{cases} \langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle \geq \gamma - \xi_i, & \forall i \in \mathcal{I}_+, \\ \langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle \leq \eta + \xi_i, & \forall i \in \mathcal{I}_-, \\ \xi_i \geq 0, & \forall i, \end{cases} \end{aligned} \tag{5.19}$$

which we call RCH–Margin. The dual of this problem, as it can be suspected, is RCH–NPP (Eq. (3.71)). Next, we will compare the KKT optimality conditions for l_1–SVC, l_1–ν–SVC and RCH–Margin with the same notation as before, so as to arrive to a figure similar to Figure 5.4. It will turn out that we can jump from any formulation to each of the other two, by just using the values of its optimal solution.

As for l_1–SVC, recalling (3.25), its KKT conditions are:

$$
\begin{aligned}
&\text{S} && \begin{cases} \overrightarrow{w}^* = \sum_{i=1}^N \alpha_i^* y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^N \alpha_i^* y_i = 0, \end{cases} \\
&\text{DF} && \begin{cases} 0 \leq \alpha_i^* \leq C, \quad i = 1, \ldots, N, \end{cases} \\
&\text{PF} && \begin{cases} y_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) \geq 1 - \xi_i^*, & i = 1, \ldots, N, \\ \xi_i^* \geq 0, & i = 1, \ldots, N, \end{cases} \\
&\text{CS} && \begin{cases} \alpha_i^* \left[y_i(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*) - 1 + \xi_i^*\right] = 0, & i = 1, \ldots, N, \\ \left(C - \alpha_i^*\right) \xi_i^* = 0, & i = 1, \ldots, N. \end{cases}
\end{aligned}
\tag{5.20}
$$

The ones for l_1–ν–SVC, by virtue of (3.31), are

$$
\begin{aligned}
&\text{S} && \begin{cases} \overrightarrow{w}' = \sum_{i=1}^N \alpha_i' y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{i=1}^N \alpha_i' y_i = 0, \\ \sum_{i=1}^N \alpha_i' = \nu, \end{cases} \\
&\text{DF} && \begin{cases} 0 \leq \alpha_i' \leq \frac{1}{N}, \quad i = 1, \ldots, N, \end{cases} \\
&\text{PF} && \begin{cases} y_i \left(\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right)\rangle + b'\right) \geq \rho' - \xi_i', & i = 1, \ldots, N, \\ \xi_i' \geq 0, & i = 1, \ldots, N, \end{cases} \\
&\text{CS} && \begin{cases} \alpha_i' \left[y_i(\langle \overrightarrow{w}', \Phi\left(\overrightarrow{x}_i\right)\rangle + b') - \rho' + \xi_i'\right] = 0, & i = 1, \ldots, N, \\ \left(\frac{1}{N} - \alpha_i'\right) \xi_i' = 0, & i = 1, \ldots, N. \end{cases}
\end{aligned}
\tag{5.21}
$$

Finally, the KKT conditions for the RCH–Margin problem (5.19) have the form

$$
\begin{aligned}
&\text{S} && \begin{cases} \overrightarrow{w}^o = \sum_{i=1}^N \alpha_i^o y_i \Phi\left(\overrightarrow{x}_i\right), \\ \sum_{\mathcal{I}_+} \alpha_i^o = \sum_{\mathcal{I}_-} \alpha_i^o = 1, \end{cases} \\
&\text{DF} && \begin{cases} 0 \leq \alpha_i^o \leq \mu, \quad i = 1, \ldots, N, \end{cases} \\
&\text{PF} && \begin{cases} \langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle \geq \gamma^o - \xi_i^o, & \forall i \in \mathcal{I}_+, \\ \langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle \leq \eta^o + \xi_i^o, & \forall i \in \mathcal{I}_-, \\ \xi_i^o \geq 0, & \forall i, \end{cases} \\
&\text{CS} && \begin{cases} \alpha_i^o \left[\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle - \gamma^o + \xi_i^o\right] = 0, & \forall i \in \mathcal{I}_+, \\ \alpha_i^o \left[\langle \overrightarrow{w}^o, \Phi\left(\overrightarrow{x}_i\right)\rangle - \eta^o - \xi_i^o\right] = 0, & \forall i \in \mathcal{I}_-, \\ \left(\mu - \alpha_i^o\right) \xi_i^o = 0, & i = 1, \ldots, N. \end{cases}
\end{aligned}
\tag{5.22}
$$

Similarly to what was done before, we can state the following:

Proposition 5.4. *The KKT conditions of l_1–ν–SVC become the KKT conditions of RCH–Margin by defining $\lambda = 2/\nu = 2/\sum_{i=1}^{N} \alpha_i'$ and making the substitutions $\overrightarrow{\alpha}^o = \lambda\overrightarrow{\alpha}'$, $\overrightarrow{w}^o = \lambda\overrightarrow{w}'$, $\gamma^o = \lambda(\rho' - b')$, $\eta^o = \lambda(-\rho' - b')$, $\mu = \lambda/N$.*

Proposition 5.5. *The KKT conditions of l_1–ν–SVC become the KKT conditions of l_1–SVC by defining $\lambda = 1/\rho' = \sum_{i=1}^{N} \alpha_i'/(\rho'\nu)$ and making the substitutions $\overrightarrow{\alpha}^* = \lambda\overrightarrow{\alpha}'$, $\overrightarrow{w}^* = \lambda\overrightarrow{w}'$, $b^* = \lambda b'$, $\xi^* = \lambda\xi'$, $C = \lambda/N$.*

Corollary 5.6 ([24], Theorem 4.4)**.** *The KKT conditions of RCH–Margin become the KKT conditions of l_1–SVC by defining $\lambda = 2/(\gamma^o - \eta^o) = \sum_{\mathcal{I}_+} \alpha_i^* = \sum_{\mathcal{I}_-} \alpha_i^*$ and making the substitutions $\overrightarrow{\alpha}^* = \lambda\overrightarrow{\alpha}^o$, $\overrightarrow{w}^* = \lambda\overrightarrow{w}^o$, $b^* = -\lambda(\gamma^o + \eta^o)/2$, $\xi^* = \lambda\xi^o$, $C = \lambda\mu$.*

Observe that the only difference between these propositions and the ones for the l_2 formulations is that the change from ν to μ, from ν to C and from μ to C are dependent on the respective values of the scale factor λ. This allows to close the circle of Figure 5.4 and to establish a one–on–one relationship among the three models, yielding Figure 5.5.

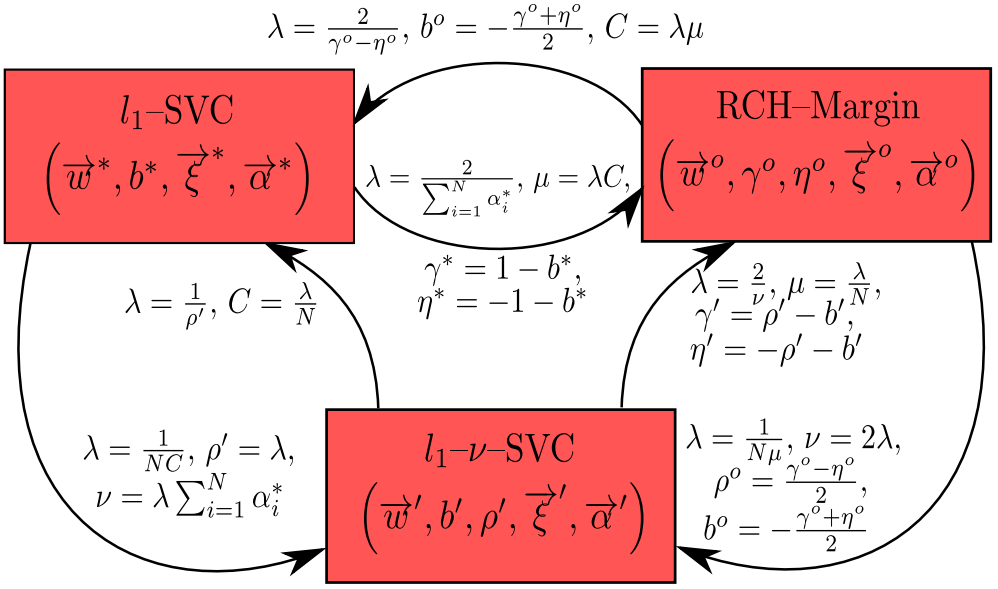

FIGURE 5.5: Relationship between the l_1–SVC, l_1–ν–SVC and RCH–Margin optimal solutions. The factor λ indicates the scale factor in the direction of its respective arrow.

It is also easy to see that the KKT conditions can be summarized in $\Delta \leq 0$, with a proper redefinition of Δ. We will illustrate this for the l_1–SVC case.

Assume again that we are solving the dual formulation in $\overrightarrow{\alpha}$, so that we automatically fulfill the S and DF conditions of (5.20). As argued in §3.1.3, we can summarize the PF and CS ones as follows:

$$
\begin{aligned}
y_i\left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &\geq 1 \quad \forall i : \alpha_i^* = 0,\\
y_i\left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &= 1 \quad \forall i : 0 < \alpha_i^* < C,\\
y_i\left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + b^*\right) &\leq 1 \quad \forall i : \alpha_i^* = C,
\end{aligned}
$$

Applying (4.7), the above can be rewritten, separating for the cases where $y_i = \pm 1$, as

$$
\begin{aligned}
y_i \nabla \mathcal{D}_i^* &\geq -b^* \quad \forall i \in \mathcal{I}_+ : \alpha_i^* < C,\\
y_i \nabla \mathcal{D}_i^* &\leq -b^* \quad \forall i \in \mathcal{I}_- : \alpha_i^* < C,\\
y_i \nabla \mathcal{D}_i^* &\leq -b^* \quad \forall i \in \mathcal{I}_+ : \alpha_i^* > 0,\\
y_i \nabla \mathcal{D}_i^* &\geq -b^* \quad \forall i \in \mathcal{I}_- : \alpha_i^* > 0.
\end{aligned}
$$

Equivalently,

$$
\min_{i:(y_i=+1\wedge\alpha_i^*<C)\vee(y_i=-1\wedge\alpha_i^*>0)} \left\{y_i \nabla \mathcal{D}_i^*\right\} \geq -b^* \geq \max_{i:(y_i=+1\wedge\alpha_i^*>0)\vee(y_i=-1\wedge\alpha_i^*<C)} \left\{y_i \nabla \mathcal{D}_i^*\right\}.
$$

Adapting (4.10) and (4.9), we can define

$$
\begin{aligned}
\mathcal{I}_L^* &= \{i : (y_i = +1 \wedge \alpha_i^* < C) \vee (y_i = -1 \wedge \alpha_i^* > 0)\},\\
\mathcal{I}_U^* &= \{i : (y_i = +1 \wedge \alpha_i^* > 0) \vee (y_i = -1 \wedge \alpha_i^* < C)\},
\end{aligned}
\tag{5.23}
$$

$$
L^* = \arg\min_{i \in \mathcal{I}_L^*} \{y_i \nabla \mathcal{D}_i\}, \qquad U^* = \arg\max_{i \in \mathcal{I}_U^*} \{y_i \nabla \mathcal{D}_i\}, \tag{5.24}
$$

together with $\Delta^* = y_{U^*} \nabla \mathcal{D}_{U^*} - y_{L^*} \nabla \mathcal{D}_{L^*}$. It follows that we can finally write

$$
\max_{\mathcal{I}_{U^*}} \left\{y_i \nabla \mathcal{D}_i^*\right\} \leq \min_{\mathcal{I}_{L^*}} \left\{y_i \nabla \mathcal{D}_i^*\right\} \quad \Rightarrow \quad \Delta^* \leq 0. \tag{5.25}
$$

As for RCH–Margin, by a similar reasoning to the one used in §5.1.2.1, the KKT conditions can be summarized in

$$
\Delta^o = \max\{\Delta_+^o, \Delta_-^o\} \leq 0, \tag{5.26}
$$

after replacing (5.14) by

$$\begin{aligned} \mathcal{I}_{L_+^o} &= \{i : (y_i = +1 \wedge \alpha_i^o < \mu)\}, \qquad \mathcal{I}_{U_+^o} = \{i : (y_i = +1 \wedge \alpha_i^o > 0)\}, \\ \mathcal{I}_{L_-^o} &= \{i : (y_i = -1 \wedge \alpha_i^o < \mu)\}, \qquad \mathcal{I}_{U_-^o} = \{i : (y_i = -1 \wedge \alpha_i^o > 0)\}. \end{aligned} \tag{5.27}$$

Operating in the same way for l_1–ν–SVC we can arrive to

$$\Delta' = \max\{\Delta'_+, \Delta'_-\} \leq 0, \tag{5.28}$$

after replacing C by $1/N$ in (5.23) and omitting the -1 in the gradient (4.7), by virtue of (3.30).

5.1.2.3 The Relationship between MNP and OC

Having seen that NPP, ν–SVC and SVC are in a sense the same problem, it is natural to think that MNP and OC should also be interrelated. Here we will see briefly that this is so.

We can define an analogous problem to (5.3):

$$\begin{aligned} \min_{\overrightarrow{w},\gamma,\overrightarrow{\xi}} \quad & \frac{1}{2}\left\|\overrightarrow{w}\right\|_2^2 - \gamma + \mu \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & \begin{cases} \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle \geq \gamma - \xi_i, & \forall i, \\ \xi_i \geq 0, & \forall i, \end{cases} \end{aligned} \tag{5.29}$$

which can be termed as OC–Margin. Its visual interpretation is that we seek to find a "support" hyperplane $\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}\right)\rangle = \gamma$ that tries to make all the samples lie in its positive halfspace. We allow for errors with slacks ξ_i, which are again weighted with the trade–off parameter μ. The compromise in the objective function is keeping these slacks at bay while trying to separate the hyperplane as much as possible from the origin, as the distance from it is $\gamma / \left\|\overrightarrow{w}\right\|_2$. An example appears in Figure 5.6.

Dualizing this problem results in the RCH–MNP problem (3.69). Comparing (5.29) and (3.63), it is clear that OC and OC–Margin are exactly the same problem after making $\rho = \gamma$ and $\mu = 1/(\nu N)$. The same happens with the l_2 version of OC, which is equivalent to the l_2 version of (5.29), whose dual is the CH–MNP problem (3.68).

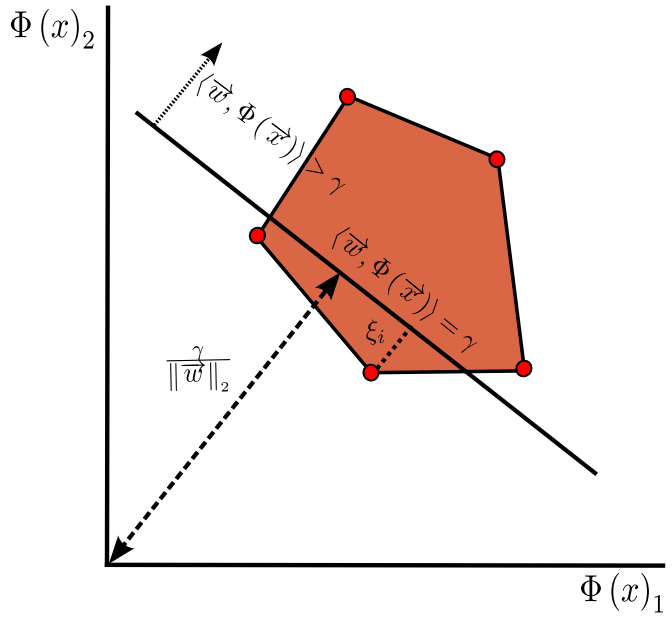

FIGURE 5.6: Illustration of the OC–Margin problem. A point that lies in the wrong halfspace of the hyperplane is highlighted with the slack variable associated to it.

5.1.3 Consequences

There are several important consequences that we can infer from the relationships detailed above, apart from confirming the visual intuition we had about the maximal margin hyperplane being determined by the closest points in the (reduced) convex hulls of the samples.

5.1.3.1 Stopping when Δ is Small is Reasonable

Maybe the most immediate consequence is that the SMO algorithm, be it for the l_1 case (Algorithm 1) or for the l_2 case, has a very natural stopping criterion when Δ is small, as we have seen that in the optimum $\Delta^* \leq 0$. In fact, this is the criterion used in LIBSVM or in SVM^{light} for stopping. Thus, we can use $\Delta > \epsilon$ in the loop of Algorithm 1, with ϵ a small value. For instance, the default value in LIBSVM and SVM^{light} is $\epsilon = 10^{-3}$. As this is an absolute value, it does not take into account the scale of $\vec{w}$, and for some problems this value has to be more finely tuned. In any case, the idea is to stop when the value of Δ is small enough, so that we are close enough to an optimum.

We mentioned in the description of SMO (§4.1.1.1) that the selection of the pair of coefficients for updating is related to the violation of the KKT conditions. Recall that first order selection (Eq. 4.9) aims to maximize Δ, so actually it chooses what is referred to as the *most violating pair (MVP)*. This is why in the literature it is also known as

the *MVP rule*. Second order selection (Eq. 4.11) indirectly tends to maximize Δ but also taking into account the dual gain, so that usually the pair selected is different than the MVP one. Whichever the selection, if we have reached the optimum, no pair can be selected.

In fact, this stopping criterion based on violation of the KKT conditions is the same criterion used in the MDM (Algorithm 5) and RCH–MDM (Algorithm 8) methods. While this stopping criterion was originally justified from a purely geometrical point of view, it turns out to be also based on violation of the optimality conditions. It might seem that GSK (Algorithm 4) and RCH–GSK (Algorithm 6) do the same, because they also stop when Δ is small. However, it is important to note that the definition of Δ for these two algorithms is different, and it is not directly related to the KKT conditions, but purely on geometry. In any case, their flavor is similar since, broadly speaking, the larger Δ is, the further we are from an optimum.

5.1.3.2 MDM is a Particular Case of SMO

Once we have understood the stopping criterion with Δ, an idea springs to mind. The MDM (RCH–MDM) and SMO for l_2–SVC (l_1–SVC) algorithms share this stopping criterion, and they both select two patterns for updating. Moreover, we designed the RCH–MDM method building on the clipping strategy adopted by SMO for l_1–SVC, and we have seen that CH–NPP (RCH–NPP) is the dual problem of CH–Margin (RCH–Margin), which in turn is a rescaled version of l_2–SVC (l_1–SVC), as illustrated in Figure 5.4 (Figure 5.5).

This suggests that actually MDM (RCH–MDM) works exactly as first order SMO for l_2–SVC (l_1–SVC), once we have switched to the CH–Margin (RCH–Margin) problem. Switching to this problem makes the pair selection be constrained to one of the classes, so that $y_L = y_U$, and the sum of coefficients $\sum_{i=1}^{N} \alpha_i$ be fixed to 2.

The following propositions show that this intuition is true. From now on, we will give preference to l_1–SVC and RCH–NPP, since these are by far the problems that have been given more attention in the last years.

Proposition 5.7. *Consider the dual of l_1–SVC in (3.24), SMO solving it with first order selection, and RCH–MDM solving problem (3.71). If both algorithms choose the same patterns for updating, and $\sum_{i=1}^{N} \alpha_i = 2$, then their updates are identical.*

Proof. Denote by L' and U' the indices of the patterns chosen by RCH–MDM, and by L, U the ones chosen by SMO. We know that $y_{L'} = y_{U'}$, because of the way RCH–MDM

selects the pair. Also, looking at Figure 5.5 we see that, if $\sum_{i=1}^{N} \alpha_i = 2$, we can switch from problem (3.24) to (3.71) by a factor of 1, so that $C = \mu$. Let us analyze separately the two possible cases.

If $y_{L'} = y_{U'} = +1$, we have $L' = L_+$ and $U' = U_+$ in (5.1). According to (4.30), and using $L' = L$, $U' = U$, we have

$$\lambda = -y_{L'} \frac{\Delta_+}{\|\overrightarrow{z}_{L'U'}\|_2^2} = \frac{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_U) - \Phi(\overrightarrow{x}_L) \rangle - (y_U - y_L)}{\|\overrightarrow{z}_{LU}\|_2^2},$$

which is nothing but (4.3). After this, the clipping strategy (5.2) tells us

$$\lambda \leftarrow \min\{\lambda, \alpha_{U'}, \mu - \alpha_{L'}\} = \min\{\lambda, \alpha_U, C - \alpha_L\},$$

which corresponds to (4.12). Finally, the update of the coefficients in (4.29) can be rewritten as

$$\begin{aligned} \alpha_{L'} \leftarrow \alpha_{L'} + \lambda \quad &\Rightarrow \quad \alpha_L \leftarrow \alpha_L + y_L \lambda, \\ \alpha_{U'} \leftarrow \alpha_{U'} - \lambda \quad &\Rightarrow \quad \alpha_U \leftarrow \alpha_U - y_U \lambda, \end{aligned}$$

so that we obtain (4.2).

When $y_{L'} = y_{U'} = -1$, we get $L' = L_-$ and $U' = U_-$ in (5.1). Using (4.30), together with $L' = U$, $U' = L$:

$$\lambda = -y_{L'} \frac{\Delta_-}{\|\overrightarrow{z}_{L'U'}\|_2^2} = \frac{\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_U) - \Phi(\overrightarrow{x}_L) \rangle - (y_U - y_L)}{\|\overrightarrow{z}_{LU}\|_2^2},$$

which is again (4.3), but this time the role of L and U is exchanged. Clipping with (5.2) now means

$$\lambda \leftarrow \min\{\lambda, \alpha_{U'}, \mu - \alpha_{L'}\} = \min\{\lambda, \alpha_L, C - \alpha_U\},$$

which corresponds to (4.12). Finally, the update of the coefficients in (4.29) can be rewritten as

$$\begin{aligned} \alpha_{L'} \leftarrow \alpha_{L'} + \lambda \quad &\Rightarrow \quad \alpha_U \leftarrow \alpha_U - y_U \lambda, \\ \alpha_{U'} \leftarrow \alpha_{U'} - \lambda \quad &\Rightarrow \quad \alpha_L \leftarrow \alpha_L + y_L \lambda, \end{aligned}$$

so that we obtain again (4.2).

□

Proposition 5.8. *Consider the dual of l_2–SVC in (3.14), SMO solving it with first order selection, and CH–MDM solving problem (3.70). If both algorithms choose the same patterns for updating, and $\sum_{i=1}^{N} \alpha_i = 2$, then their updates are identical.*

Proof. The proof follows the same scheme as in Proposition 5.7. The l_2 version of SMO has not been explained in detail, but it is straightforward to see that the only change with respect to its l_1 version is in switching (4.10) to

$$\begin{aligned} \mathcal{I}_L &= \{i : (y_i = +1) \vee (y_i = -1 \wedge \alpha_i > 0)\}, \\ \mathcal{I}_U &= \{i : (y_i = +1 \wedge \alpha_i > 0) \vee (y_i = -1)\}, \end{aligned}$$

and the clipping part from (4.12) to

$$\begin{aligned} \lambda &\leftarrow \min\{\lambda, \alpha_L\} \quad \text{if} \quad y_L = -1, \\ \lambda &\leftarrow \min\{\lambda, \alpha_U\} \quad \text{if} \quad y_U = +1. \end{aligned}$$

□

What these results show is that RCH–MDM (CH–MDM) can be regarded as a particular case of the first order SMO version for l_1 (l_2)–SVC, where the patterns selected for updating are restricted to belong to the same class, and such that the sum of the Lagrange coefficients remains equal to 2. In other words, MDM applies the same rationale than SMO, tailoring it to the particular NPP problem that it wants to solve.

It can also be shown that the original MDM algorithm for solving the MNP problem is equivalent to the SMO version for solving OC, using the relationship outlined in §5.1.2.3.

5.1.3.3 RCH–MDM is Sound and Faster than RCH–GSK

Next, we will conduct a series of experiments aimed to compare our proposed RCH–MDM method against RCH–GSK. We will also confirm in these experiments that RCH–MDM is a particular case of l_1–SMO.

To do so, G. Rätsch's benchmark datasets [1] have been used, namely these 13 binary classification datasets: *Titanic* (T), *Heart* (H), *Diabetes* (D), *Breast Cancer* (BC), *Thyroid* (Th), *Flare* (F), *Splice* (S), *Image* (I), *German* (G), *Banana* (B), *Twonorm* (Tw),

Ringnorm (R) and *Waveform* (W). These datasets have been widely used in the literature. The main advantage of using the benchmark [1] is that for each problem there are train–test splits already available (100 for each dataset except for *Image* and *Splice*, where only 20 splits are given).

For this series of experiments we will use the Gaussian kernel of Definition 2.37, which is the most common choice for a kernel. Another advantage of this benchmark is that it provides us, for each dataset, with C and σ values for the Gaussian kernel that result in good classification performances for l_1–SVC. Table 5.1 contains these values, as well as the training and test sizes of the splits and the dimensionality of the input patterns.

SET	DIM	N_{Tr}	N_{Te}	C	σ^2
T	3	150	2051	5.0	2.0
H	13	170	100	3.2	120.0
BC	9	200	77	15.2	50.0
TH	5	140	75	10.0	3.0
D	8	468	300	1.0	20.0
F	9	666	400	1.0	30.0
S	60	1000	2175	1000.0	70.0
I	18	1300	1010	500.0	30.0
G	20	700	300	3.2	55.0
B	2	400	4900	316.2	1.0
R	20	400	7000	10^9	10.0
TW	20	400	7000	3.2	40.0
W	21	400	4600	1.0	20.0

TABLE 5.1: Number of dimensions, training and test patterns, together with the suggested hyperparameter values for a Gaussian kernel for the datasets in [1].

The three algorithms are programmed in C and tested on a machine with 4 GB of RAM and 4 Intel Core 2 Quad 2.83–GHz processors, running under Ubuntu 10.04 LTS. In order to test them under uniform conditions, we have to deal with two main difficulties:

- Using equivalent parameters for the different formulations.
- Stopping the algorithms following similar criteria.

For the first one, we will work with the l_1–SVC problem, since it is for that formulation that we have the parameters of Table 5.1. Using Corollary 5.6, we can switch from that formulation to RCH–Margin/NPP. However, note that this change of formulation is dependent on the optimal solution for l_1–SVC. Thus, even if we have single values of C in Table 5.1 for each dataset, in every train–test split the optimal solution is different, and as a result we will not have single values for μ. The σ value remains the same in all the formulations, since it is specific to the Gaussian kernel.

What we do then is choosing the C values of Table 5.1, train an l_1–SVM with SMO, and once it finishes for each training–test pair, subsequently solve RCH–NPP with RCH–MDM and RCH–GSK, using a μ value computed according to Corollary 5.6. The case of l_1–ν–SVC is omitted, because the SMO algorithm adapted for this problem is formally equivalent to RCH–MDM, since the sum of the Lagrange coefficients is fixed to ν, the pair selected must belong to the same class, and the dual has exactly the same form. Therefore, the number of iterations and selections performed are virtually the same as the ones obtained by RCH–MDM solving RCH–NPP.

For this to work reasonably well, it is important that l_1–SMO stops close to an optimal solution. Strictly speaking, it is only at the very optimum that the equivalence of Corollary 5.6 holds. Even if we cannot in practice reach this optimum, using an approximate solution is, in general, enough. Whereas l_1–SMO and RCH–MDM have been described as stopping when the quantity Δ is less than a given tolerance ϵ in §5.1.3.1, we have to keep in mind that this is not exactly the case for RCH–GSK, where an alternative Δ is defined which does not correspond directly to KKT violation.

Hence, we stop the algorithms using a different idea. Because we are solving the dual formulations, we know that the respective duals are being minimized. Even if the optima are not reached, we asymptotically tend to them (convergence will be treated in Chapter 6). Then, typically what happens is that in the first iterations we descend quickly, while later in the algorithm we progress much more slowly, eventually getting almost stagnant.

This observation, together with the fact that the duals we are solving are different, suggests that we use the following alternative criterion; stopping at iteration t when it holds that

$$\frac{|\mathcal{D}^{t-1} - \mathcal{D}^t|}{|\mathcal{D}^{t-1}|} < \epsilon.$$

The above fraction measures the relative change in the dual function. It is important that this change be relative, because the magnitudes involved in each dual are different. For instance, the dual value in (3.71) cannot be negative, as it is a norm. On the other hand, the dual value in (3.24) cannot be positive, because $\overrightarrow{\alpha} = \overrightarrow{0}$ is a feasible (and trivial) solution and the algorithms are initialized in this point. Hence the absolute values in the formula.

Note that l_1–SMO and RCH–MDM tend to maximize the change in the dual function because of the way the pair is chosen (especially in second order selection), so this criterion somewhat favors RCH–GSK. In any case, we will see that the differences are

Set	l_1–SMO	RCH–MDM	RCH–GSK
T	22.6±2.4	22.9±4.6	22.7±2.7
H	15.9±3.2	16.1±3.2	16.1±3.2
BC	26.3±4.6	29.4±4.6	31.2±5.1
Th	4.4±2.2	4.8±2.4	4.3±2.1
D	23.5±1.7	23.9±2.0	23.9±2.0
F	32.8±1.7	32.9±1.6	32.8±1.7
S	10.8±0.6	10.9±0.7	10.9±0.7
I	3.4±0.8	4.8±2.4	27.5±11.7
G	23.6±2.1	24.3±2.1	25.0±3.2
B	12.1±2.6	17.9±13.0	21.1±12.3
R	1.7±0.1	1.7±0.1	1.7±0.1
Tw	3.0±0.2	3.0±0.2	3.0±0.2
W	9.9±0.4	9.9±0.4	9.9±0.4

Table 5.2: Averages and standard deviations of the test errors (in %) obtained with the l_1–SMO, RCH–MDM and RCH–GSK algorithms.

still very remarkable. The value of ϵ is set to 10^{-6}, so that the l_1–SMO solution is accurate enough.

Table 5.2 shows the performances given by the obtained models in the test splits. These performances are in general similar under a Wilcoxon rank test at the 10% level, but in some cases RCH–MDM and RCH–GSK give significantly worse models, most notably in datasets *Breast Cancer*, *Image* and *Banana*.

This effect can be explained by two causes. The first one is that, once convergence is attained, the way of calculating the bias in l_1–SVC and RCH–NPP is different: whereas l_1–SVC calculates it as an average value based on the KKT conditions (Eq. (3.26)), RCH–NPP makes the classification hyperplane bisect the segment joining the two closest points in the reduced convex hulls, as was done for standard convex hulls in Figure 5.2. While for the hard–margin case these biases are identical, when slacks are present they are, in general, different, even if the weight vectors obtained are identical [47]. The second one, as we will see below, is that for some problems we need a more accurate ϵ.

Table 5.3 shows that the number of SVs obtained by l_1–SMO and RCH–MDM is also similar, which is a strong (although not definite) evidence that the models are quite the same, especially when we also take into account the similarity in the classification performances of Table 5.2. In fact, there are again significant differences in two of the three datasets described above (*Image* and *Banana*), but also in *Titanic*, *Flare* and *Splice*.

The cases of *Titanic* and *Flare* are rather special, because they present patterns which are identical but belong to different classes. This situation provokes that there are many

Set	l_1–SMO	RCH–MDM	RCH–GSK
T	68.2±9.9	113.6±8.8	144.7±9.4
H	82.5±5.4	82.5±5.4	93.7±5.5
BC	116.7±6.1	115.3±6.0	149.6±7.0
Th	25.1±5.7	25.1±5.7	76.0±42.0
D	264.9±7.5	265.2±7.4	280.8±7.2
F	474.8±12.4	509.4±9.9	600.3±23.2
S	602.1±27.9	627.6±13.5	943.1±135.5
I	173.8±11.3	138.5±9.2	356.0±19.3
G	400.8±20.4	408.9±11.4	443.8±13.4
B	114.7±21.7	95.7±14.7	173.1±50.6
R	150.3±10.8	152.2±10.7	400.0±0.0
Tw	74.4±6.1	75.0±6.2	98.4±7.4
W	160.0±9.0	161.0±8.8	193.2±10.7

Table 5.3: Averages and standard deviations of the number of support vectors obtained with the l_1–SMO, RCH–MDM and RCH–GSK algorithms.

equivalent yet internally different models (e.g. with two such patterns we can let their coefficients be 0 or C, and the model is still the same).

Regarding *Image* and *Banana*, the number of SVs in RCH–MDM is less than for l_1–SMO. This indicates that the value of ϵ used is not accurate enough, because the switch from C to μ is not as sound as it should be. Under normal circumstances, l_1–SMO has slightly less SVs than RCH–MDM due to the initialization of the coefficients: RCH–MDM is initialized in the barycenters of the reduced convex hulls (hence, every point is a SV initially), whereas l_1–SMO is initialized at $\overrightarrow{\alpha} = \overrightarrow{0}$. The barycenters are chosen because it is a safe way to ensure that we lie inside the hulls. SMO could also be initialized in such a way, but it does not make much sense to do so, as there is no geometry involved, and the number of kernel operations required for that initialization is much greater ($\mathcal{O}(N^2)$). Hence, if we use a stricter ϵ we can tackle this issue, although the training times are significantly larger (in [93] this was shown for an $\epsilon = 10^{-10}$).

In dataset *Splice*, similar accuracies are obtained in spite of the different number of SVs. This happens as well with RCH–GSK, which has by far more SVs than the other two methods. Looking at the rest of Table 5.3, it is seen that RCH–GSK always has considerably more SVs. This is another drawback of the RCH–GSK method when compared to RCH–MDM. By the nature of GSK's update (4.36), the coefficient of a point which should not be a SV at the optimum can only be asymptotically shrunk by the factor $1 - \lambda$. As a consequence, a large number of iterations are needed to shrink a coefficient so much that it can be considered zero. Observe for instance that in *Ringnorm* the algorithm is not able to remove a single SV after the initialization in the barycenters, and also that in the datasets where RCH–MDM showed a poor performance, RCH–GSK

shows an even poorer one (in *Image* and *Banana*). In contrast, RCH–MDM can remove a wrong SV in a single operation, provided that $\lambda = \alpha_U$ in (4.29).

Even if the chosen tolerance guarantees that we are close enough to the optimum, RCH–GSK usually needs more iterations to converge than RCH–MDM and l_1–SMO. Furthermore, its iterations are much more computationally expensive than for the other two algorithms. This is seen very clearly in Table 5.4 and Table 5.5. There are a few datasets (*Diabetes*, *German*, *Banana* and *Waveform*) for which RCH–GSK needs less iterations than RCH–MDM but, even so, it has to compute many more kernel operations, specifically $\mathcal{O}(\lfloor 1/\mu \rfloor N)$ versus $\mathcal{O}(2N)$.

Set	l_1–SMO	RCH–MDM	RCH–GSK
T	116.9±18.5	123.3±17.6	9552.8±29754.9
H	108.8±16.2	211.8±15.6	241.1±53.3
BC	494.7±115.7	748.4±213.1	1500.4±427.8
Th	134.6±42.3	224.1±34.0	999.3±259.6
D	236.1±24.8	537.7±19.0	336.1±56.7
F	309.7±32.1	720.3±80.1	863.9±349.6
S	955.2±185.2	1208.4±258.1	14093.0±2641.1
I	5944.9±1177.3	12601.1±1858.2	16913.8±2342.1
G	604.8±138.8	1065.9±195.5	940.2±307.1
B	7201.9±1937.5	35809.3±70265.2	34688.5±32101.3
R	207.0±13.1	368.1±16.6	5518.8±269.7
Tw	180.7±21.2	440.3±16.5	765.4±95.9
W	176.9±19.7	430.0±14.2	404.1±58.0

Table 5.4: Averages and standard deviations of the number of iterations needed with the l_1–SMO, RCH–MDM and RCH–GSK algorithms.

It can also be seen how RCH–MDM always needs more iterations to converge than l_1–SMO. This is natural, since the pair is restricted to be chosen in the same class in RCH–MDM, whereas l_1–SMO can pick two patterns from different classes. The different initialization is important, too. As both algorithms have a complexity of $\mathcal{O}(2N)$ kernel operations per iteration, the ratio between the number of iterations and the number of kernel operations in both methods is virtually the same.

To sum up, it is clear that RCH–MDM is a better algorithm than RCH–GSK for solving the RCH–NPP problem. It usually needs less iterations to converge, and each iteration is much cheaper in RCH–MDM than in RCH–GSK, especially when the reduction coefficient μ is small. On the other hand, l_1–SMO is better than RCH–MDM to solve l_1–SVC.

The l_2 case was also considered in [93], where we compared l_2–SMO and CH–MDM. The same phenomenon was observed: SMO is slightly faster due to its additional degree

Set	l_1–SMO	RCH–MDM	RCH–GSK
T	35.7±5.7	37.4±5.3	54086.0±99403.6
H	37.5±5.6	72.7±5.3	2241.4±504.6
BC	200.3±46.8	301.6±85.9	21099.9±4408.0
Th	38.3±12.0	63.4±9.6	773.4±211.5
D	222.3±23.4	504.9±17.8	30342.5±5125.0
F	414.2±42.9	961.6±106.9	222222.4±85514.8
S	1915.3±371.3	2420.3±517.0	17164.0±9838.6
I	15488.8±3066.6	32800.7±4836.9	887780.8±94226.6
G	849.8±194.9	1495.4±274.2	170476.3±39074.7
B	5801.4±1559.5	28754.8±56423.0	498675.2±399278.6
R	166.6±10.5	295.6±13.4	2218.5±108.4
Tw	145.5±17.1	353.5±13.3	6967.9±981.3
W	142.5±15.8	345.3±11.4	13563.8±2264.1

Table 5.5: Averages and standard deviations of the number of kernel operations (in thousands) needed with the l_1–SMO, RCH–MDM and RCH–GSK algorithms.

of freedom when choosing the working set and its simpler initialization. The models obtained were more homogeneous than in the l_1 case because the equivalence is direct, making $C = \mu$ (see Figure 5.4), so that we do not need to change from C to μ in every train–test split. We observed that the GSK method is in general slower than CH–MDM, because it again can only remove asymptotically the wrong SVs. However, its cost per iteration is only $\mathcal{O}(N)$, so that the overhead of GSK is not so remarkable as the one of RCH–GSK we have seen here.

5.1.3.4 The Interest of Solving NPP for Classification

At this point, it may seem that the equivalence we have established between NPP and SVC is only interesting from a theoretical point of view. After all, the (RCH–)MDM algorithm, which is the state–of–the–art for solving (R)CH–NPP, is a particular and restricted case of l_2–SMO (l_1–SMO), so that in practice it is slower. Then we could conclude that, if we are to solve SVC, the best option is to do so with the SMO algorithm, and forget about any geometrical interpretation.

Nevertheless, this is a hasty conclusion. Having a geometrical interpretation is always interesting, because we can gain new insights from the visual feedback that the MDM algorithm gives. Not only is this possible, but also the other way round: we can try to use some of the ideas of SMO in MDM, so as to obtain a better geometric algorithm if we are to solve directly an NPP problem, for instance in a robotics context [50].

Some examples of insights that have been explored are given next, although its detailed description is beyond the scope of this thesis:

- An example of an idea of SMO that can be applied in MDM is second order pair selection. In general second order works better than first order for SMO, but in MDM the improvement, in most of the cases, turns out to be only a marginal one.

- In the spirit of SVM–Light, we can think of an extension of MDM that chooses more than two patterns for updating. The case of four patterns (the two from the positive class, and the two from the negative one) was described in [94].

- Second order has a better performance than first order in SMO because it can be observed that, in first order, some pairs are repeatedly chosen as the algorithm progresses. In the MDM case it can be seen that, as a result, the algorithm zigzags inside the hulls, instead of choosing a more sensible direction. Building on this observation, we proposed, jointly with Á. Barbero, a basic *cycle–breaking* strategy for MDM and SMO that tries to identify cycles in the pair selection and build an update direction that is a combination of the update directions that were used in the cycle, with the hope that this direction is better. This strategy, though simple, produces improvements in both algorithms [95, 96], especially when first order is used.

- As an extension of the *cycle–breaking* strategy, one can devise a *momentum* term that keeps track of the average direction used in the most recent iterations of SMO/MDM. This momentum term can be combined with the standard update at a given point. It can be shown that the optimal combination is analytically solvable [64]. An example of application for LS–SVC and LS–SVR has been recently given in [97]. Noticeable savings are observed.

- Hybrid algorithms with both a geometrical and an algebraical interpretation can also be designed: a basic example of such an algorithm is Rosen's method in [98]. Here, the gradient of the dual is multiplied by a matrix such that the resulting direction is guaranteed to be feasible and improving. It turns out that, for the SVC case, the calculation of this matrix does not need any inversion, contrary to the general case.

These examples show that the interplay between geometric and decomposition algorithms can be exploited to improve the performance of both types of algorithms, so it is not a good idea to focus just on a particular kind of methods, but to keep in mind their mutual relationship.

5.2 A General Formulation for all Problems

Once we have seen that l_2–SVC, l_2–ν–SVC and CH–NPP are basically the same problem, that the same can be said about l_1–SVC, l_1–ν–SVC and RCH–NPP, and that OC and MNP are equivalent, a natural question that arises is: can we go a step further and unify all these formulations under a general problem that encompasses classification, regression and OC? We will see in this section that in fact we can.

5.2.1 Derivation

In order to come up with a general formulation, it is useful to identify common points in the primal and dual formulations of Chapters 3 and 4. Let us start with the objective function:

- The term $(1/2)\|\overrightarrow{w}\|_2^2$ is always present.
- There is always a term with the sum of the slacks, which is multiplied by a trade–off parameter. This parameter is equal to C in SVC, SVR, LS–SVC and LS–SVR, and it is assigned $1/N$ in ν–SVC, $1/(N\nu)$ in OC(–Margin), C/N in ν–SVR, and μ in (R)CH–Margin.
- Depending on these slacks we have two types of machines: if the slacks are squared, the trade–off parameter is divided by 2 for ease of differentiability and in the dual the Lagrangian coefficients are not bounded from above. On the other hand, if we are using an l_1–loss, the slacks are not squared and are constrained to be non–negative. As a result, in the dual the Lagrangian coefficients are bounded from above by the trade–off parameter.
- ν–SVMs also present a term $-\nu\rho$ in classification and $C\nu\epsilon$ in regression, where C and ν are fixed beforehand.
- OC presents the additional term $-\rho$. However, ν can be different from 1, so this is not exactly the same as for ν–SVMs.
- The objective function is always minimized with respect to the weight vector $\overrightarrow{w}$, the bias b (except in OC, where there is no bias) and the slacks $\overrightarrow{\xi}$. In the case of ν–SVMs, it is also minimized with respect to ρ in classification, and ϵ in regression.
- In all regression cases (except for LS–SVR), the slack variables are duplicated, so we also have a vector $\overrightarrow{\xi}'$ with respect to which the minimization is carried out.

Now let us examine the constraints:

- As we have pointed above, the non–negativity constraints of the slacks only appear when these are not squared.
- In SVC, the constraints are of the form $y_i(\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle + b) \geq 1 - \xi_i$.
- LS–SVC has constraints of the same form, but with equalities instead of inequalities.
- ν–SVC has the same constraints than SVC, with ρ replacing 1.
- LS–SVR has the same constraints than LS–SVC, but with y_i replacing 1.
- OC has analogous constraints than ν–SVC, after removal of y_i and the bias term b.
- In ϵ–SVR and ν–SVR the constraints are duplicated so as to ensure that the point is not too much above or below the ϵ–tube.
- In the dual, the lower bound of the Lagrangian coefficients is always 0, except in LS–SVC and LS–SVR, where they are unconstrained.

Considering all this, LS–SVMs are the models that most differ from the others: instead of inequality constraints they use equality constraints, and their Lagrangian coefficients are unconstrained. In this section, we will study a general framework that covers all models except LS–SVC and LS–SVR. These will be incorporated in §5.2.3 by means of an even more general framework.

Thus, leaving LS–SVMs temporarily outside of our study, in order to arrive to a general formulation we should take care of:

- Including the term $(1/2)\|\overrightarrow{w}\|_2^2$ in the objective function with no changes.
- Encompassing the l_1 and l_2 losses in the slacks by means of a general term.
- Including an additional term in the objective function with a variable over which the function is minimized.
- Designing a scheme such that the constraints are implicitly duplicated for ϵ–SVR and ν–SVR.
- Removing implicitly the bias term for the OC case.

To this aim, we can think of the following primal problem:

$$\begin{aligned} \min_{\overrightarrow{w},b,\overrightarrow{\xi},\rho} \quad & \frac{1}{2}\left\|\overrightarrow{w}\right\|_2^2 + r\rho + \frac{u}{p}\sum_i \xi_i^p \\ \text{s.t.} \quad & \begin{cases} t_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb\right) \geq q_i - \xi_i - z\rho, & \forall i, \\ \xi_i \geq 0, & \forall i. \end{cases} \end{aligned} \tag{5.30}$$

Let us explain briefly the role of the different variables. Here, $\overrightarrow{w}$, b, $\overrightarrow{\xi}$ and ρ are the variables upon which the problem is minimized. Their meaning should be clear from the different SVM formulations studied so far.

As for p, s, r, z, u, t_i and q_i, these are prefixed scalars. The value p is equal to 1 or 2: if $p = 1$ we are in an l_1 scheme, and the sum of the values of the slacks is calculated, while if $p = 2$ we are using the l_2 norm and the slacks are squared in the sum.

The scalar s is used to remove effectively the bias from the problem, by making $s = 0$. In a similar fashion, r is set to 0 when we do not want an extra term with ρ in the objective function, and we choose $z = 0$ if this parameter ρ should not appear in the constraints. Regarding u, we will see that it will have two different roles in the dual problem, depending on whether $p = 1$ or $p = 2$. In the former case, it will be the upper bound for the Lagrangian coefficients, while in the latter case it will modify the kernel function being used. Finally, t_i and q_i should be chosen so that we obtain the constraints of each particular problem. Depending on the particular problem being included, i will range from 1 to N (in classification models) or from 1 to $2N$ (in regression ones).

Aside from p being restricted to either 1 or 2, some additional assumptions on the variable values are used. No infinite values are allowed in any scalar, and it is also assumed that $u > 0$. Additionally, it is implicitly assumed that the values being chosen guarantee that problem (5.30) is feasible. Thus, we assume that there will always be at least one finite solution $\left(\overrightarrow{w}^*, b^*, \overrightarrow{\xi}^*, \rho^*\right)$ for the problem. As a counterexample, consider for instance that we choose $z = 0$ and $r < 0$. Then there is no finite solution, as the objective function can be made arbitrarily small by making ρ arbitrarily large.

It should also be noted that the non–negativity constraints on $\overrightarrow{\xi}$ are redundant when $p = 2$. The reasoning for this is similar to what we said for l_2–SVC in §3.1.2. Let us assume that $\overrightarrow{w}$, b and ρ are temporarily fixed. If $t_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb) > q_i - z\rho$ we will not take $\xi_i < 0$, but $\xi_i = 0$, because this choice minimizes the objective function (recall that $u > 0$). Conversely, if $t_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb) \leq q_i - z\rho$ we take $\xi_i = q_i - z\rho - t_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb)$, so that $\xi_i \geq 0$.

With these observations in mind, we can easily see that SVC is a particular instance of (5.30):

Proposition 5.9. *The primal* (5.30) *becomes the* l_1*–SVC* ($p = 1$) *and* l_2*–SVC* ($p = 2$) *primals, by making* $u = C$, $r = 0$, $s = 1$, $z = 0$, $t_i = y_i, i = 1, \ldots, N$ *and* $q_i = +1, i = 1, \ldots, N$.

Proof. Direct substitution in (5.30) makes the objective function become

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{p}\sum_{i=1}^{N}\xi_i^p,$$

which is minimized over $\overrightarrow{w}$, b and $\overrightarrow{\xi}$, since ρ can be removed from the problem. The constraints obtained are of the form

$$\begin{aligned} y_i(\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle + b) \geq 1 - \xi_i, \quad & i = 1, \ldots, N, \\ \xi_i \geq 0, \quad & i = 1, \ldots, N. \end{aligned}$$

When $p = 2$ the constraints $\xi_i \geq 0$ are redundant, yielding thus (3.11). When $p = 1$ they are necessary, obtaining (3.21). □

The same holds for ν–SVC:

Proposition 5.10. *The primal* (5.30) *becomes the* l_1*–*ν*–SVC* ($p = 1$) *and* l_2*–*ν*–SVC* ($p = 2$) *primals, by making* $u = 1/N$, $r = -\nu$, $s = 1$, $z = -1$, $t_i = y_i, i = 1, \ldots, N$ *and* $q_i = 0, i = 1, \ldots, N$.

Proof. Direct substitution in (5.30) makes the objective function become

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 - \nu\rho + \frac{1}{pN}\sum_{i=1}^{N}\xi_i^p,$$

which is minimized over $\overrightarrow{w}$, b, $\overrightarrow{\xi}$ and ρ. The constraints are of the form

$$\begin{aligned} y_i(\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle + b) \geq \rho - \xi_i, \quad & i = 1, \ldots, N, \\ \xi_i \geq 0, \quad & i = 1, \ldots, N. \end{aligned}$$

Again, the latter are redundant when $p = 2$, so (5.30) becomes (3.33), as the constraint $\rho \geq 0$ in this problem was also seen to be redundant in §3.1.4. This also happens when $p = 1$, so we have that (5.30) is transformed into (3.27). □

OC is also easily seen to be encompassed:

Proposition 5.11. *The primal* (5.30) *becomes the* l_1*–OC* ($p = 1$) *and* l_2*–OC* ($p = 2$) *primals, by making* $u = 1/(\nu N)$, $r = -1$, $s = 0$, $z = -1$, $t_i = +1, i = 1, \ldots, N$ *and* $q_i = 0, i = 1, \ldots, N$.

Proof. Direct substitution in (5.30) makes the objective function become

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 - \rho + \frac{1}{p\nu N}\sum_{i=1}^{N}\xi_i^p,$$

which is minimized over $\overrightarrow{w}$, $\overrightarrow{\xi}$ and ρ. We do not minimize with respect to b, since it can be removed from the problem. The constraints obtained have the form

$$\begin{aligned}\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle \geq \rho - \xi_i, \quad i = 1, \ldots, N,\\ \xi_i \geq 0, \quad i = 1, \ldots, N,\end{aligned}$$

so that $p = 1$ yields (3.63). When $p = 2$ the non–negativity constraints on $\overrightarrow{\xi}$ are unnecessary, which gives l_2–OC. □

The regression cases are a bit trickier, as the samples have to be duplicated to cover the constraints. Let us start with ϵ–SVR:

Proposition 5.12. *The primal* (5.30) *becomes the* l_1*–*ϵ*–SVR* ($p = 1$) *and* l_2*–*ϵ*–SVR* ($p = 2$) *primals, by making* $u = C$, $r = 0$, $s = 1$, $z = 0$, $t_i = +1, i = 1, \ldots, N$, $t_{N+i} = -1, i = 1, \ldots, N$, $q_i = y_i - \epsilon, i = 1, \ldots, N$, $q_{N+i} = -y_i - \epsilon, i = 1, \ldots, N$, *and* $\overrightarrow{x}_{N+i} = \overrightarrow{x}_i, i = 1, \ldots, N$.

Proof. Direct substitution in (5.30) makes the objective function become

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{p}\sum_{i=1}^{2N}\xi_i^p,$$

which is minimized over $\overrightarrow{w}$, b and $\overrightarrow{\xi}$. Splitting the $2N$ constraints in two sets, according to the values in the proposition, yields:

$$\begin{aligned} y_i - (\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle + b) &\leq \epsilon + \xi_i, \quad i = 1, \ldots, N, \\ (\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_{N+i}) \rangle + b) - y_{N+i} &\leq \epsilon + \xi_{N+i}, \quad i = 1, \ldots, N, \\ \xi_i &\geq 0, \quad i = 1, \ldots, N, \\ \xi_{N+i} &\geq 0, \quad i = 1, \ldots, N. \end{aligned}$$

Substituting $p = 1$ clearly yields (3.48), where the only difference is that instead of a $2N$–dimensional slack vector we have two N–dimensional vectors $\overrightarrow{\xi}$ and $\overrightarrow{\xi}'$.

When $p = 2$ the last two sets of constraints are redundant, so we obtain (3.44). □

The inclusion of ν–SVR is similar:

Proposition 5.13. *The primal (5.30) becomes the l_1–ν–SVR ($p = 1$) and l_2–ν–SVR ($p = 2$) primals, by making $u = C/N$, $r = C\nu$, $s = 1$, $z = 1$, $t_i = +1, i = 1, \ldots, N$, $t_{N+i} = -1, i = 1, \ldots, N$, $q_i = y_i, i = 1, \ldots, N$, $q_{N+i} = -y_i, i = 1, \ldots, N$, and $\overrightarrow{x}_{N+i} = \overrightarrow{x}_i, i = 1, \ldots, N$.*

Proof. Direct substitution in (5.30) makes the objective function become

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 + C\left(\nu\rho + \frac{1}{pN}\sum_{i=1}^{2N}\xi_i^p\right),$$

which is minimized over $\overrightarrow{w}$, b, $\overrightarrow{\xi}$ and ρ. This last parameter plays the role of ϵ. As in Proposition 5.12, $\overrightarrow{\xi}$ is a 2N–dimensional vector, and the constraints become:

$$\begin{aligned} y_i - (\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle + b) &\leq \rho + \xi_i, \quad i = 1, \ldots, N, \\ (\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_{N+i}) \rangle + b) - y_{N+i} &\leq \rho + \xi_{N+i}, \quad i = 1, \ldots, N, \\ \xi_i &\geq 0, \quad i = 1, \ldots, N, \\ \xi_{N+i} &\geq 0, \quad i = 1, \ldots, N. \end{aligned}$$

so that substituting $p = 1$ clearly yields (3.53), as we mentioned in §3.2.3 that the constraint $\epsilon \geq 0$ was redundant. When $p = 2$ this is also true, and the non–negativity constraints on $\overrightarrow{\xi}$ are not needed, so that we obtain the primal of l_2–ν–SVR. □

Problems CH–Margin and RCH–Margin can also be included by exploiting the relationships described in §5.1.2 and §5.1.2.2.

Proposition 5.14. *The primal (5.30) becomes the RCH–Margin ($p = 1$) and CH–Margin ($p = 2$) problems, by making $u = \mu$, $r = 2$, $s = 1$, $z = 1$, $t_i = y_i, i = 1, \ldots, N$ and $q_i = 0, i = 1, \ldots, N$.*

Proof. Direct substitution in (5.30) makes the objective function become

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 + 2\rho + \frac{\mu}{p}\sum_{i=1}^{N}\xi_i^p,$$

which is minimized over $\overrightarrow{w}$, b, $\overrightarrow{\xi}$ and ρ. The constraints are

$$\begin{aligned} y_i(\langle\overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle + b) \geq -\rho - \xi_i, \quad & i = 1, \ldots, N, \\ \xi_i \geq 0, \quad & i = 1, \ldots, N. \end{aligned}$$

Using $\gamma = -\rho - b$, $\eta = \rho - b$, so that $\rho = (\eta - \gamma)/2$, the objective function can be written as

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 - \gamma + \eta + \frac{\mu}{p}\sum_{i=1}^{N}\xi_i^p,$$

and the constraints as

$$\begin{aligned} \langle\overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle \geq \gamma - \xi_i, \quad & \forall i \in \mathcal{I}_+, \\ \langle\overrightarrow{w}, \Phi(\overrightarrow{x}_i)\rangle \leq \eta + \xi_i, \quad & \forall i \in \mathcal{I}_-, \\ \xi_i \geq 0, \quad & \forall i, \end{aligned}$$

obtaining directly (5.19) with $p = 1$, and (5.3) with $p = 2$ after removal of the unnecessary constraints $\xi_i \geq 0$. □

5.2.2 A General SMO Algorithm

Having encompassed all the primal formulations in a single primal, it should also be the case that all the dual formulations can be included in a single dual, which is indeed

the dual of our general primal. Here we will derive this general dual and see that everything fits. Furthermore, if we define an SMO algorithm for the resulting dual with the same rationale than in §4.1.1.1, we obtain a general SMO algorithm that encompasses as particular cases the SMO variants of each specific formulation, and MDM in the geometric formulations due to Propositions 5.7 and 5.8.

Starting with the l_1 case, the Lagrangian of (5.30) when $p = 1$ is

$$\begin{aligned}\mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \rho, \overrightarrow{\alpha}, \overrightarrow{\delta}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + r\rho + u\sum_i \xi_i \\ - \sum_i \alpha_i \left[t_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb) - q_i + \xi_i + z\rho\right] - \sum_i \delta_i \xi_i.\end{aligned} \tag{5.31}$$

According to Definitions 2.60 and 2.61, the dual problem is

$$\begin{aligned}&\max_{\overrightarrow{\alpha}, \overrightarrow{\delta}} \quad \inf_{\overrightarrow{w}, b, \overrightarrow{\xi}, \rho} \quad \mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \rho, \overrightarrow{\alpha}, \overrightarrow{\delta}\right) \\ &\text{s.t.} \quad \begin{cases} \alpha_i \geq 0 & \forall i, \\ \delta_i \geq 0 & \forall i. \end{cases}\end{aligned} \tag{5.32}$$

The above infimum can be automatically found by equaling to zero the gradient of the Lagrangian (5.31), producing

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \overrightarrow{w}} = \overrightarrow{w} - \textstyle\sum_i \alpha_i t_i \Phi\left(\overrightarrow{x}_i\right) = 0 \quad &\Rightarrow \quad \overrightarrow{w} = \textstyle\sum_i \alpha_i t_i \Phi\left(\overrightarrow{x}_i\right), \\ \frac{\partial \mathcal{L}}{\partial b} = -s\textstyle\sum_i \alpha_i t_i = 0 \quad &\Rightarrow \quad s\textstyle\sum_i \alpha_i t_i = 0, \\ \frac{\partial \mathcal{L}}{\partial \rho} = r - z\textstyle\sum_i \alpha_i = 0 \quad &\Rightarrow \quad z\textstyle\sum_i \alpha_i = r, \\ \frac{\partial \mathcal{L}}{\partial \xi_i} = u - \alpha_i - \delta_i = 0 \quad &\Rightarrow \quad 0 \leq \alpha_i \leq u.\end{aligned} \tag{5.33}$$

Substitution of (5.33) in (5.31) finally gives the dual

$$\begin{aligned}&\min_{\overrightarrow{\alpha}} \quad \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j t_i t_j K_{ij} - \sum_i \alpha_i q_i \\ &\text{s.t.} \quad \begin{cases} 0 \leq \alpha_i \leq u, & \forall i, \\ s\textstyle\sum_i \alpha_i t_i = 0, \\ z\textstyle\sum_i \alpha_i = r. \end{cases}\end{aligned} \tag{5.34}$$

The derivation of the l_2 case goes very much along the lines of the l_1 case just seen. As the constraints $\xi_i \geq 0$ are redundant when $p = 2$, there is no need of Lagrange multipliers $\overrightarrow{\delta}$. The Lagrangian of (5.30) now becomes

$$\begin{aligned}\mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \rho, \overrightarrow{\alpha}\right) &= \frac{1}{2}\|\overrightarrow{w}\|_2^2 + r\rho + \frac{u}{2}\sum_i \xi_i^2 \\ &- \sum_i \alpha_i \left[t_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb) - q_i + \xi_i + z\rho\right], \end{aligned} \tag{5.35}$$

whereas the dual problem is

$$\begin{aligned} \max_{\overrightarrow{\alpha}} \quad & \inf_{\overrightarrow{w}, b, \overrightarrow{\xi}, \rho} \quad \mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\xi}, \rho, \overrightarrow{\alpha}\right) \\ \text{s.t.} \quad & \alpha_i \geq 0 \ \ \forall i. \end{aligned} \tag{5.36}$$

Since the only difference in (5.35) with (5.31) is in the terms with $\overrightarrow{\xi}$, (5.33) still applies for all variables aside from $\overrightarrow{\xi}$. Equaling the corresponding partial derivative to 0 gives instead

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = u\xi_i - \alpha_i = 0 \quad \Rightarrow \quad \alpha_i = u\xi_i. \tag{5.37}$$

Substituting (5.37) and the rest of (5.33) in (5.35) gives the dual

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j t_i t_j K_{ij} + \frac{1}{2u}\|\overrightarrow{\alpha}\|_2^2 - \sum_i \alpha_i q_i \\ \text{s.t.} \quad & \begin{cases} 0 \leq \alpha_i, & \forall i, \\ s\sum_i \alpha_i t_i = 0, \\ z\sum_i \alpha_i = r, \end{cases} \end{aligned}$$

or, equivalently,

$$\begin{aligned} \min_{\vec{\alpha}} \quad & \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j t_i t_j \left(K_{ij} + \frac{\delta_{ij}}{u}\right) - \sum_i \alpha_i q_i \\ \text{s.t.} \quad & \begin{cases} 0 \leq \alpha_i, & \forall i, \\ s\sum_i \alpha_i t_i = 0, \\ z\sum_i \alpha_i = r, \end{cases} \end{aligned} \tag{5.38}$$

where we use the fact that $t_i = \pm 1$ in all the formulations considered.

It is worth observing that u acts as an upper bound for the Lagrangian coefficients in (5.34), whereas in (5.38) these coefficients are not bounded from above, and u only takes part in the modification of the kernel function. When $s = 0$ the constraint $s\sum_i \alpha_i y_i = 0$ is trivial and can be removed from both duals. Similarly, when $z = r = 0$, the constraint $z\sum_i \alpha_i = r$ can also be removed.

It is straightforward to see then that performing the same substitutions than in Propositions 5.9–5.14 yields the duals of the respective primals. For the l_1 cases, RCH–MNP and RCH–NPP, it is the dual (5.34) that has to be used, whereas for the l_2 cases, CH–MNP and CH–NPP it is (5.38).

For ease of reference, we summarize in Table 5.6 the values of the different variables, with $i = 1, \ldots, N$.

Formulation	u	r	s	z	t_i	q_i	t_{N+i}	q_{N+i}	$\vec{x}_{N+i}$
SVC	C	0	$+1$	0	y_i	$+1$	—	—	—
ν–SVC	$1/N$	$-\nu$	$+1$	-1	y_i	0	—	—	—
OC/MNP	$1/(\nu N)$	-1	0	-1	$+1$	0	—	—	—
ϵ–SVR	C	0	$+1$	0	$+1$	$y_i - \epsilon$	-1	$-y_i - \epsilon$	$\vec{x}_i$
ν–SVR	C/N	$C\nu$	$+1$	$+1$	$+1$	y_i	-1	$-y_i$	$\vec{x}_i$
(R)CH–Margin/NPP	μ	$+2$	$+1$	$+1$	y_i	0	—	—	—

Table 5.6: Variables used in the general formulation to include the different specific formulations.

With this table in mind, we can now devise a general SMO algorithm along the lines of §4.1.1.1. Recall that we only want to modify two Lagrangian coefficients α_L and α_U according to the scheme

$$\begin{aligned} \alpha_L &\leftarrow \alpha_L + \lambda_L, \\ \alpha_U &\leftarrow \alpha_U + \lambda_U, \\ \alpha_i &\leftarrow \alpha_i, \quad \forall i \neq L, U, \end{aligned}$$

where λ denotes the advance factor. Now we have to distinguish among three different cases, according to the equality constraints of (5.34) or (5.38):

1. Both constraints are non–trivial: this is the case when $s \neq 0$, $r \neq 0$, $z \neq 0$, that is, in ν–SVC, ν–SVR and (R)CH–Margin/NPP.
2. The constraint $s \sum_i \alpha_i t_i = 0$ is trivial, and $z \sum_i \alpha_i = r$ is non–trivial: this happens when $s = 0$, $r \neq 0$, $z \neq 0$. Table 5.6 says that OC is the only formulation with such values.
3. The constraint $z \sum_i \alpha_i = r$ is trivial, and $s \sum_i \alpha_i t_i = 0$ is non–trivial: for this to hold we must have $s \neq 0$, $r = 0$, $z = 0$, which is the case of SVC and ϵ–SVR.

The remaining case (both constraints being trivial with $s = r = z = 0$) would correspond to minimizing the dual function only subject to box constraints, but this does not happen in the formulations considered.

Clearly, if the constraint $s \sum_i \alpha_i t_i = 0$ is non–trivial, we must have $t_L \lambda_L + t_U \lambda_U = 0$ and, therefore, $\lambda_U = -t_L t_U \lambda_L$. Additionally, if the constraint $z \sum_i \alpha_i = r$ is non–trivial we must also have $\lambda_L + \lambda_U = 0$, so that $\lambda_U = -\lambda_L$.

Thus, the above three cases respectively translate, in terms of λ_L and λ_U, into:

1. $\lambda_U = -t_L t_U \lambda_L$ and $\lambda_L = -\lambda_U$, so $t_L = t_U$.
2. $\lambda_U = -\lambda_L$. Note anyway that in the OC case we have $t_L = t_U$ because $t_i = +1 \; \forall i$.
3. $\lambda_U = -t_L t_U \lambda_L$, but it is not required that $t_L = t_U$.

Therefore, we can consider without loss of generality that $\lambda_U = -t_L t_U \lambda_L$ in all three cases, keeping in mind that in the first case we have to force that $t_L = t_U$.

Applying (5.37) and what was said in §3.1.2, our l_2 case can be viewed as a generalization of the hard–margin case [43], where we modify the weight vector to

$$\widetilde{\overrightarrow{w}} = \left(\overrightarrow{w}, \sqrt{u}\, \overrightarrow{\xi} \right) = \left(\overrightarrow{w}, \frac{\overrightarrow{\alpha}}{\sqrt{u}} \right), \tag{5.39}$$

and the feature map to

$$\widetilde{\Phi}\left(\overrightarrow{x}_i \right) = \left(\Phi\left(\overrightarrow{x}_i \right), \frac{t_i}{\sqrt{u}} \overrightarrow{e}_i \right), \tag{5.40}$$

which cause that the new kernel function is

$$\widetilde{k}\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) = k\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) + \frac{\delta_{ij}}{u} \tag{5.41}$$

and also that

$$\left\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}\left(\overrightarrow{x}_i\right) \right\rangle = \left\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + t_i \xi_i = \left\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + t_i \frac{\alpha_i}{u}, \tag{5.42}$$

so that $t_i \left(\left\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}\left(\overrightarrow{x}_i\right) \right\rangle + sb \right) = t_i \left(\left\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + sb \right) + \xi_i$ in the constraints of (5.30).

In the sequel we will use the tilde symbol to mean the modified weight vector (5.39) and kernel function (5.41) if we are working with the l_2 dual (5.38). If we are considering the l_1 dual (5.34), the tilde symbol does not have any effect, so $\overrightarrow{\widetilde{w}} = \overrightarrow{w}$ and $\widetilde{\Phi}(\cdot) = \Phi(\cdot)$.

We are thus able to express simultaneously (5.34) and (5.38). Writing the subsequent optimization problem in matrix notation we get:

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \ \mathcal{D}(\overrightarrow{\alpha}) \quad &= \quad \frac{1}{2} \left\langle \overrightarrow{\alpha}, \widetilde{\Omega} \overrightarrow{\alpha} \right\rangle - \left\langle \overrightarrow{\alpha}, \overrightarrow{q} \right\rangle \\ \text{s.t.} \quad &\begin{cases} \overrightarrow{0} \leq \overrightarrow{\alpha} \leq \overrightarrow{v}, \\ s \left\langle \overrightarrow{\alpha}, \overrightarrow{t} \right\rangle = 0, \\ z \left\langle \overrightarrow{\alpha}, \overrightarrow{1} \right\rangle = r, \end{cases} \end{aligned} \tag{5.43}$$

where $\widetilde{\Omega}_{ij} = t_i t_j \widetilde{k}\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right)$. We also introduced the vector $\overrightarrow{v}$, which is equal to $\overrightarrow{u}$ for (5.34) and to $+\overrightarrow{\infty}$ for (5.38). Although we mentioned previously that we were not going to use infinite values, here it is useful to have a single dual problem that allows to derive the algorithm at once, without the need of separating the l_1 and l_2 cases. Note as well that $\left\langle \overrightarrow{\alpha}, \widetilde{\Omega} \overrightarrow{\alpha} \right\rangle = \left\| \overrightarrow{\widetilde{w}} \right\|_2^2$.

Because of our analysis of the three different situations we can encounter, we can express the update in $\overrightarrow{\alpha}$ as

$$\overrightarrow{\alpha} \leftarrow \overrightarrow{\alpha} + \overrightarrow{\lambda}, \tag{5.44}$$

where $\overrightarrow{\lambda}$ is a vector with every component set to 0 except for the L–th (set to λ_L) and the U–th (set to $-t_U t_L \lambda_L$) ones. This means that the update in the dual can be expressed as

$$\begin{aligned} \mathcal{D} &\leftarrow \frac{1}{2}\left\langle \overrightarrow{\alpha}+\overrightarrow{\lambda}, \widetilde{\Omega}\left(\overrightarrow{\alpha}+\overrightarrow{\lambda}\right)\right\rangle - \left\langle \overrightarrow{\alpha}+\overrightarrow{\lambda}, \overrightarrow{q}\right\rangle \\ &= \mathcal{D} + \left\langle \overrightarrow{\lambda}, \widetilde{\Omega}\overrightarrow{\alpha}\right\rangle + \frac{1}{2}\left\langle \overrightarrow{\lambda}, \widetilde{\Omega}\overrightarrow{\lambda}\right\rangle - \left\langle \overrightarrow{\lambda}, \overrightarrow{q}\right\rangle . \end{aligned} \tag{5.45}$$

It is straightforward to see by virtue of (5.41) and (5.42) that in this formula we obtain

$$\left\langle \overrightarrow{\lambda}, \widetilde{\Omega}\overrightarrow{\alpha}\right\rangle = \lambda_L t_L \left\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}\left(\overrightarrow{x}_L\right) - \widetilde{\Phi}\left(\overrightarrow{x}_U\right)\right\rangle, \qquad \left\langle \overrightarrow{\lambda}, \widetilde{\Omega}\overrightarrow{\lambda}\right\rangle = \lambda_L^2 \left\| \widetilde{\Phi}\left(\overrightarrow{x}_L\right) - \widetilde{\Phi}\left(\overrightarrow{x}_U\right)\right\|_2^2 .$$

As a result, the dual update (5.45) is expressible as a function $\psi\left(\lambda_L\right)$, namely

$$\begin{aligned} \mathcal{D} \leftarrow\ & \mathcal{D}^t + \frac{1}{2}\lambda_L^2 \left\| \widetilde{\Phi}\left(\overrightarrow{x}_L\right) - \widetilde{\Phi}\left(\overrightarrow{x}_U\right)\right\|_2^2 + \lambda_L t_L \left\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}\left(\overrightarrow{x}_L\right) - \widetilde{\Phi}\left(\overrightarrow{x}_U\right)\right\rangle - \\ & \lambda_L \left(q_L - t_U t_L q_U\right). \end{aligned}$$

Differentiating $\psi(\lambda_L)$ and equaling to zero yield the unconstrained optimal λ_L

$$\hat{\lambda}_L = \frac{t_L\left(\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}\left(\overrightarrow{x}_U\right) - \widetilde{\Phi}\left(\overrightarrow{x}_L\right)\rangle - \left(t_U q_U - t_L q_L\right)\right)}{\left\| \widetilde{\Phi}\left(\overrightarrow{x}_U\right) - \widetilde{\Phi}\left(\overrightarrow{x}_L\right)\right\|_2^2} = t_L \frac{\Delta}{\left\| \overrightarrow{\widetilde{z}}_{UL}\right\|_2^2} = t_L \overline{\lambda}, \tag{5.46}$$

where now we define $\overrightarrow{\widetilde{z}}_{UL} = \widetilde{\Phi}\left(\overrightarrow{x}_U\right) - \widetilde{\Phi}\left(\overrightarrow{x}_L\right)$ and $\Delta = \langle \overrightarrow{\widetilde{w}}, \overrightarrow{\widetilde{z}}_{UL}\rangle - \left(t_U q_U - t_L q_L\right)$.

Because the i–th element of the gradient of (5.43) is given by

$$\nabla \mathcal{D}_i = \left(\widetilde{\Omega}\overrightarrow{\alpha}\right)_i - q_i = t_i \sum_j \alpha_j t_j \widetilde{k}\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right) - q_i = t_i \left\langle \overrightarrow{\widetilde{w}}, \widetilde{\Phi}(\overrightarrow{x}_i)\right\rangle - q_i, \tag{5.47}$$

Δ can also be rewritten as

$$\Delta \;=\; t_U \nabla \mathcal{D}_U - t_L \nabla \mathcal{D}_L. \tag{5.48}$$

Plugging (5.46) into (5.45) gives

$$\mathcal{D} \;\leftarrow\; \mathcal{D} - \frac{\Delta^2}{2 \left\| \overrightarrow{\tilde{z}}_{UL} \right\|_2^2}, \tag{5.49}$$

which is analogous to (4.5) for l_1–SMO.

Hence, it is immediate to adapt first–order selection to choose the updating pair: in view of (5.46), if $t_L = +1$ we get a positive value for $\hat{\lambda}_L$, so it must be the case that $\alpha_L < v$. Similarly, if $y_L = -1$ we must have $\alpha_L > 0$. Since $\lambda_U = -t_U t_L \lambda_L$, it is also required that $\alpha_U > 0$ if $t_U = +1$, whereas if $t_U = -1$ we should have $\alpha_U < v$. Then the selection becomes

$$L = \arg\min_{i \in \mathcal{I}_L} \{t_i \nabla \mathcal{D}_i\}, \qquad U = \arg\max_{i \in \mathcal{I}_U} \{t_i \nabla \mathcal{D}_i\}, \tag{5.50}$$

where the index sets $\mathcal{I}_L$ and $\mathcal{I}_U$ are defined as

$$\begin{array}{rcl} \mathcal{I}_L & = & \{i : (t_i = +1 \wedge \alpha_i < v) \vee (t_i = -1 \wedge \alpha_i > 0)\}, \\ \mathcal{I}_U & = & \{i : (t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1 \wedge \alpha_i < v)\}. \end{array} \tag{5.51}$$

Note that in the l_2 case $v = +\infty$, so actually every α_i is less than v. This matches with the index sets discussed in the proof of Proposition 5.8.

However, picking (5.46) can make us violate the box constraints. By a similar argument to the one used for l_1–SMO, four different clips are possible:

$$\begin{array}{rcll} \overline{\lambda} & \leftarrow & \min\{\lambda, \alpha_L\} & \text{if} \quad t_L = -1, \\ \overline{\lambda} & \leftarrow & \min\{\lambda, v - \alpha_L\} & \text{if} \quad t_L = +1, \\ \overline{\lambda} & \leftarrow & \min\{\lambda, v - \alpha_U\} & \text{if} \quad t_U = -1, \\ \overline{\lambda} & \leftarrow & \min\{\lambda, \alpha_U\} & \text{if} \quad t_U = +1. \end{array} \tag{5.52}$$

Note that in the l_2 case the two clips with v obviously have no effect.

Regarding the stopping criterion, we have to check the KKT conditions. When $p = 2$, these are easily seen to be

$$\begin{array}{ll} \text{S} & \begin{cases} \overrightarrow{w}^* = \sum_i \alpha_i^* t_i \Phi\left(\overrightarrow{x}_i\right), \\ s \sum_i \alpha_i^* t_i = 0, \\ z \sum_i \alpha_i^* = r, \end{cases} \\ \text{DF} & \begin{cases} 0 \leq \alpha_i^* \quad \forall i, \end{cases} \\ \text{PF} & \begin{cases} t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + s b^*\right) \geq q_i - \frac{\alpha_i^*}{u} - z\rho^* \quad \forall i, \end{cases} \\ \text{CS} & \begin{cases} \alpha_i^* \left[t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + s b^*\right) - q_i + \frac{\alpha_i^*}{u} + z\rho^* \right] = 0 \quad \forall i. \end{cases} \end{array} \tag{5.53}$$

Separating the cases $\alpha_i^* = 0$ and $\alpha_i^* > 0$, this means

$$\begin{aligned} t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + s b^*\right) + \tfrac{\alpha_i^*}{u} &\geq q_i - z\rho^* \quad \forall i : \alpha_i^* = 0, \\ t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + s b^*\right) + \tfrac{\alpha_i^*}{u} &= q_i - z\rho^* \quad \forall i : \alpha_i^* > 0, \end{aligned}$$

which, using (5.47), implies

$$\begin{aligned} t_i \nabla \mathcal{D}_i^* &\geq -s b^* - z\rho^* \quad \forall i \in \mathcal{I}_+, \\ t_i \nabla \mathcal{D}_i^* &\leq -s b^* + z\rho^* \quad \forall i \in \mathcal{I}_-, \\ t_i \nabla \mathcal{D}_i^* &= -s b^* - z\rho^* \quad \forall i \in \mathcal{I}_+ : \alpha_i^* > 0, \\ t_i \nabla \mathcal{D}_i^* &= -s b^* + z\rho^* \quad \forall i \in \mathcal{I}_- : \alpha_i^* > 0, \end{aligned}$$

There are two possible cases now. If either z or ρ^* are zero, we can combine the above so that

$$\min_{i: y_i = +1 \vee (y_i = -1 \wedge \alpha_i^* > 0)} \left\{ t_i \nabla \mathcal{D}_i^* \right\} \geq -s b^* \geq \max_{i: (y_i = +1 \wedge \alpha_i^* > 0) \vee y_i = -1} \left\{ t_i \nabla \mathcal{D}_i^* \right\}.$$

Using now that $v = +\infty$ for this case, the definition of Δ (5.48) and the selection scheme of SMO given by (5.50) and (5.51), we can summarize the KKT conditions in the single equation

$$\max_{\mathcal{I}_{U^*}} \left\{ t_i \nabla \mathcal{D}_i^* \right\} \leq \min_{\mathcal{I}_{L^*}} \left\{ t_i \nabla \mathcal{D}_i^* \right\} \quad \Rightarrow \quad \Delta^* \leq 0. \tag{5.54}$$

On the other hand, if z and ρ^* are not zero, we cannot combine the positive and negative patterns and have to treat them separately:

$$\begin{aligned} \min_{i\in\mathcal{I}_+}\{t_i\nabla\mathcal{D}_i^*\} &\geq \max_{i\in\mathcal{I}_+:\alpha_i^*>0}\{t_i\nabla\mathcal{D}_i^*\}, \\ \min_{i\in\mathcal{I}_-:\alpha_i^*>0}\{t_i\nabla\mathcal{D}_i^*\} &\geq \max_{i\in\mathcal{I}_-}\{t_i\nabla\mathcal{D}_i^*\}, \end{aligned}$$

where $\mathcal{I}_\pm = \{i : t_i = \pm 1\}$.

If $z \neq 0$, we already know that SMO chooses L and U separately for $t_i = \pm 1$ according to

$$\begin{aligned} L_+ &= \arg\min_{i\in\mathcal{I}_+:\alpha_i<v}\{t_i\nabla\mathcal{D}_i\}, & U_+ &= \arg\max_{i\in\mathcal{I}_+:\alpha_i>0}\{t_i\nabla\mathcal{D}_i\}, \\ L_- &= \arg\min_{i\in\mathcal{I}_-:\alpha_i>0}\{t_i\nabla\mathcal{D}_i\}, & U_- &= \arg\max_{i\in\mathcal{I}_-:\alpha_i<v}\{t_i\nabla\mathcal{D}_i\}, \end{aligned} \tag{5.55}$$

which results in two different values $\Delta_\pm$, as happened for CH–Margin and l_2–ν–SVC in §5.1.2.1. Since $v = \infty$, the points chosen at an optimum are

$$\begin{aligned} L_+^* &= \arg\min_{i\in\mathcal{I}_+}\{t_i\nabla\mathcal{D}_i^*\}, & U_+^* &= \arg\max_{i\in\mathcal{I}_+:\alpha_i>0}\{t_i\nabla\mathcal{D}_i^*\}, \\ L_-^* &= \arg\min_{i\in\mathcal{I}_-:\alpha_i>0}\{t_i\nabla\mathcal{D}_i^*\}, & U_-^* &= \arg\max_{i\in\mathcal{I}_-}\{t_i\nabla\mathcal{D}_i^*\}, \end{aligned} \tag{5.56}$$

and the KKT conditions can be summarized in

$$\Delta^* = \max\{\Delta_+^*, \Delta_-^*\} \leq 0. \tag{5.57}$$

Hence, it is important to keep in mind that there are two modes of operation in the general SMO algorithm. If $z \neq 0$, the indices chosen have to belong to the same class, and the class finally chosen is the one whose $\Delta_\pm$ is largest. This largest $\Delta_\pm$ is non–positive at every optimum. Otherwise, if $z = 0$ the indices can be chosen from different classes, giving a unique Δ which is also non–positive at every optimum.

When $p = 1$, the KKT optimality conditions can be seen to be

$$
\begin{array}{ll}
\text{S} & \begin{cases} \overrightarrow{w}^* = \sum_i \alpha_i^* t_i \Phi\left(\overrightarrow{x}_i\right), \\ s \sum_i \alpha_i^* t_i = 0, \\ z \sum_i \alpha_i^* = r, \end{cases} \\
\text{DF} & \begin{cases} 0 \leq \alpha_i^* \leq u \quad \forall i, \end{cases} \\
\text{PF} & \begin{cases} t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + s b^*\right) \geq q_i - \xi_i^* - z\rho^* \quad \forall i, \\ \xi_i^* \geq 0 \qquad\qquad\qquad\qquad\qquad\quad \forall i, \end{cases} \\
\text{CS} & \begin{cases} \left(u - \alpha_i^*\right) \xi_i^* = 0 \qquad\qquad\qquad\qquad\qquad\quad \forall i, \\ \alpha_i^* \left[t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + s b^*\right) - q_i + \xi_i^* + z\rho^*\right] = 0 \quad \forall i, \end{cases}
\end{array}
\tag{5.58}
$$

Separating the cases $\alpha_i^* = 0$, $0 < \alpha_i^* < u$ and $\alpha_i^* = u$, this means

$$
\begin{aligned}
t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + s b^*\right) &\geq q_i - z\rho^* \quad \forall i : \alpha_i^* = 0, \\
t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + s b^*\right) &= q_i - z\rho^* \quad \forall i : 0 < \alpha_i^* < u, \\
t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + s b^*\right) &\leq q_i - z\rho^* \quad \forall i : \alpha_i^* = u,
\end{aligned}
$$

which, using (5.47), implies

$$
\begin{aligned}
t_i \nabla \mathcal{D}_i^* &\geq -s b^* - z\rho^* \quad \forall i \in \mathcal{I}_+ : \alpha_i^* < C, \\
t_i \nabla \mathcal{D}_i^* &\leq -s b^* + z\rho^* \quad \forall i \in \mathcal{I}_- : \alpha_i^* < C, \\
t_i \nabla \mathcal{D}_i^* &\leq -s b^* - z\rho^* \quad \forall i \in \mathcal{I}_+ : \alpha_i^* > 0, \\
t_i \nabla \mathcal{D}_i^* &\geq -s b^* + z\rho^* \quad \forall i \in \mathcal{I}_- : \alpha_i^* > 0,
\end{aligned}
$$

Again, there are two possible cases depending on the values of z and ρ^*. If any of them is zero, we can combine the above so that

$$
\min_{i:(y_i=+1 \wedge \alpha_i^* < C) \vee (y_i=-1 \wedge \alpha_i^* > 0)} \left\{t_i \nabla \mathcal{D}_i^*\right\} \geq -s b^* \geq \max_{i:(y_i=+1 \wedge \alpha_i^* > 0) \vee (y_i=-1 \wedge \alpha_i^* < C)} \left\{t_i \nabla \mathcal{D}_i^*\right\},
$$

so that, in view of the definition of Δ (5.48) and the selection scheme of SMO in (5.50) and (5.51) with $v = u$, we can summarize the KKT conditions in the single equation

$$
\max_{\mathcal{I}_{U^*}} \left\{t_i \nabla \mathcal{D}_i^*\right\} \leq \min_{\mathcal{I}_{L^*}} \left\{t_i \nabla \mathcal{D}_i^*\right\} \quad \Rightarrow \quad \Delta^* \leq 0. \tag{5.59}
$$

Algorithm 9 General SMO algorithm

while $\Delta \geq \epsilon$ **do**
  Select working set (L, U) with first order (Eq. (5.50)) or second order (Eq. (5.61)).
  Compute $\overline{\lambda} = \frac{\Delta}{\|\overrightarrow{\tilde{z}}_{UL}\|_2^2}$ following (5.46).
  Clip λ if necessary using (5.52).
  Compute λ_L with (5.46) and $\lambda_U = -t_L t_U \lambda_L$.
  Update $\overrightarrow{\alpha}$ with (5.44).
end while

If neither of them is zero, by a similar reasoning to the l_2 case we know that the selection is performed separately in each class and that the KKT conditions can be summarized in

$$\Delta^* = \max\{\Delta_+^*, \Delta_-^*\} \leq 0. \tag{5.60}$$

Although the above discussion was made with first order selection, the same is applicable to second order, as no pair can be chosen if $\Delta \leq 0$. This selection in the general case amounts to

$$L = \arg\min_{i \in \mathcal{I}_L} \{t_i \nabla \mathcal{D}_i\}, \qquad U = \arg\max_{i \in \mathcal{I}_U, t_i \nabla \mathcal{D}_i > t_L \nabla \mathcal{D}_L} \left\{ \frac{(t_i \nabla \mathcal{D}_i - t_L \nabla \mathcal{D}_L)^2}{\left\| \overrightarrow{\tilde{z}}_{iL} \right\|_2^2} \right\}. \tag{5.61}$$

The pseudocode of the general SMO algorithm appears in Algorithm 9.

It can be easily seen that performing the same substitutions than in Propositions 5.9–5.14 yields the different versions of the SMO/MDM algorithms.

We conclude this section with a table that summarizes how the coefficients are chosen in all the formulations considered, using the values given in Table 5.6. The respective selections appear in Table 5.7. Note that for finding L in l_2–OC there are no constraints in the search, as there is only one class and no upper bound. The last column shows in which mode SMO is operating: if $t_L = t_U$ the selection is performed separately in each class, and Δ is defined as the maximum of Δ_+ and Δ_-, whereas if it is not required that $t_L = t_U$ the selection of the patterns is global, giving directly a value for Δ.

Next, we explain how we can further generalize the framework in §5.2.1 so as to include also LS–SVMs.

Problem	$\mathcal{I}_L$	$\mathcal{I}_U$	$t_L = t_U$
l_1–SVC	$(t_i = +1 \wedge \alpha_i < C) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1 \wedge \alpha_i < C)$	⊠
l_2–SVC	$(t_i = +1) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1)$	⊠
l_1–ν–SVC	$\left(t_i = +1 \wedge \alpha_i < \frac{1}{N}\right) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee \left(t_i = -1 \wedge \alpha_i < \frac{1}{N}\right)$	☑
l_2–ν–SVC	$(t_i = +1) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1)$	☑
l_1–OC	$\alpha_i < \frac{1}{\nu N}$	$\alpha_i > 0$	—
l_2–OC	—	$\alpha_i > 0$	—
l_1–ϵ–SVR	$(t_i = +1 \wedge \alpha_i < C) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1 \wedge \alpha_i < C)$	⊠
l_2–ϵ–SVR	$(t_i = +1) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1)$	⊠
l_1–ν–SVR	$\left(t_i = +1 \wedge \alpha_i < \frac{C}{N}\right) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee \left(t_i = -1 \wedge \alpha_i < \frac{C}{N}\right)$	☑
l_2–ν–SVR	$(t_i = +1) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1)$	☑
RCH	$(t_i = +1 \wedge \alpha_i < \mu) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1 \wedge \alpha_i < \mu)$	☑
CH	$(t_i = +1) \vee (t_i = -1 \wedge \alpha_i > 0)$	$(t_i = +1 \wedge \alpha_i > 0) \vee (t_i = -1)$	☑

Table 5.7: Selections performed by Algorithm 9 on the different formulations considered so far. The last column says whether it is enforced or not that $t_L = t_U$ (or whether it is implicit because all t_i are identical).

5.2.3 Inclusion of LS–SVMs

As we mentioned in §5.2.1, the main difficulty for including LS–SVMs is that they use equality instead of inequality constraints. As a byproduct of this fact, their Lagrange multipliers in the dual formulation are unconstrained. For the rest of formulations, these multipliers are always non–negative.

In order to tackle this, we propose to further generalize (5.30) in the following way:

$$\begin{aligned} \min_{\overrightarrow{w},b,\overrightarrow{\tau},\overrightarrow{\xi},\rho} \quad & \frac{1}{2}\|\overrightarrow{w}\|_2^2 + r\rho - l\sum_i \tau_i + \frac{u}{p}\|\overrightarrow{\xi}\|_p^p \\ \text{s.t.} \quad & \begin{cases} t_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb\right) = q_i + \tau_i - \xi_i - z\rho, & \forall i, \\ \tau_i \geq 0, \quad \forall i, \end{cases} \end{aligned} \tag{5.62}$$

Observe that there are two differences in (5.62) with respect to (5.30). The first one is the inclusion of the non–negative variables τ_i. These variables are weighted in the objective function by a scalar l. It is assumed that $l < u$, and they apprear in the right–hand side of the constraints, which are now equalities instead of inequalities. The key is that these equality constraints can effectively become inequality constraints, because it is also forced that $\tau_i \geq 0$.

For the particular case where $l = 0$, the term $\sum_i \tau_i$ has no influence in the objective function, so it does not matter what the particular values of τ_i are; we can eliminate them from the problem and rewrite the constraints as $t_i\left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb\right) \geq q_i - \xi_i - z\rho$, recovering thus (5.30).

The second difference is in the treatment of the slacks ξ_i. It is not forced anymore that $\xi_i \geq 0$ because in the objective function it is the l_p–norm of the slack vector $\overrightarrow{\xi}$ which is being minimized, instead of $\sum_i \xi_i^p$. For $p = 2$ both terms coincide, but for $p = 1$ we are minimizing $\sum_i |\xi_i|$, not $\sum_i \xi_i$. In any case, the discussion in §5.2.1 shows us that the solution of problems (5.62) and (5.30) is identical, since we are not going to pick any negative ξ_i.

Therefore, problem (5.62) can be viewed as a generalization of problem (5.30) that allows for values for the parameter l different from 0. It is then clear that Propositions 5.9–5.14 are still valid for (5.62), once we fix $l = 0$.

This fact gives us the intuition that l is going to act as the lower bound in the Lagrange multipliers. In fact, this turns out to be the case. Analogously to (5.31), the Lagrangian of (5.62) for the l_1 case is

$$\begin{aligned}\mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\tau}, \overrightarrow{\xi}, \rho, \overrightarrow{\alpha}, \overrightarrow{\delta}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + r\rho - l\sum_i \tau_i + u\|\overrightarrow{\xi}\|_1 \\ - \sum_i \alpha_i \left[t_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb) - q_i - \tau_i + \xi_i + z\rho\right] - \sum_i \delta_i \tau_i. \end{aligned} \tag{5.63}$$

However, there is a difficulty here, because the l_1–norm $\|\overrightarrow{\xi}\|_1$ is not differentiable. As a result, we cannot derive the dual in the same way than in §5.2.2. Fortunately, we can resort to the notion of convex conjugate in Definition 2.65.

The dual problem is now

$$\begin{aligned} \max_{\overrightarrow{\alpha}, \overrightarrow{\delta}} \quad & \inf_{\overrightarrow{w}, b, \overrightarrow{\xi}, \rho} \quad \mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\tau}, \overrightarrow{\xi}, \rho, \overrightarrow{\alpha}, \overrightarrow{\delta}\right) \\ \text{s.t.} \quad & \delta_i \geq 0 \ \ \forall i. \end{aligned} \tag{5.64}$$

Comparing this to (5.32), note that the α_i coefficients are no longer forced to be non–negative, as they are associated now to equality instead of inequality constraints.

The above infimum can be split in the different variables, yielding

$$\begin{gathered} \inf_{\overrightarrow{w}} \left\{\tfrac{1}{2}\|\overrightarrow{w}\|_2^2 - \textstyle\sum_i \alpha_i t_i \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle\right\} + \inf_b \left\{-sb \textstyle\sum_i \alpha_i t_i\right\} + \textstyle\sum_i \alpha_i q_i + \\ \inf_\rho \left\{\rho\left(r - z\textstyle\sum_i \alpha_i\right)\right\} + \inf_{\overrightarrow{\tau}} \left\{\textstyle\sum_i \tau_i\left(\alpha_i - \delta_i - l\right)\right\} + \inf_{\overrightarrow{\xi}} \left\{u\|\overrightarrow{\xi}\|_1 - \textstyle\sum_i \alpha_i \xi_i\right\}. \end{gathered}$$

For variables $\overrightarrow{w}$, b and ρ, the infima can be obtained by differentiating and equaling to 0, which gives again the first three equalities of (5.33).

As for $\overrightarrow{\tau}$, this can also be differentiated and equaled to 0:

$$\frac{\partial \mathcal{L}}{\partial \tau_i} = \alpha_i - \delta_i - l = 0 \quad \Rightarrow \quad \alpha_i \geq l. \tag{5.65}$$

As $u > 0$, the infimum with respect to $\overrightarrow{\xi}$ can be rewritten as

$$\inf_{\overrightarrow{\xi}} \left\{ u \left\| \overrightarrow{\xi} \right\|_1 - \sum_i \alpha_i \xi_i \right\} = - \sup_{\overrightarrow{\xi}} \left\{ \left\langle \frac{1}{u} \overrightarrow{\alpha}, u \overrightarrow{\xi} \right\rangle - \left\| u \overrightarrow{\xi} \right\|_1 \right\}.$$

Applying now Definition 2.65, we can identify $\overrightarrow{x} = u\overrightarrow{\xi}$, $f(\overrightarrow{x}) = \|\overrightarrow{x}\|_1$ and $\overrightarrow{x}^* = (1/u)\overrightarrow{\alpha}$. Since the convex conjugate of $f(\overrightarrow{x}) = \|\overrightarrow{x}\|_1$ is given by

$$f^*(\overrightarrow{x}^*) = \begin{cases} 0 & \text{if } \|\overrightarrow{x}\|_\infty \leq 1, \\ +\infty & \text{otherwise} \end{cases}$$

[31], we can write the infimum as

$$- \sup_{\overrightarrow{\xi}} \left\{ \left\langle \frac{1}{u} \overrightarrow{\alpha}, u \overrightarrow{\xi} \right\rangle - \left\| u \overrightarrow{\xi} \right\|_1 \right\} = \begin{cases} 0 & \text{if } \left\| \frac{1}{u} \overrightarrow{\alpha} \right\|_\infty \leq 1, \\ -\infty & \text{otherwise,} \end{cases}$$

or, equivalently,

$$- \sup_{\overrightarrow{\xi}} \left\{ \left\langle \frac{1}{u} \overrightarrow{\alpha}, u \overrightarrow{\xi} \right\rangle - \left\| u \overrightarrow{\xi} \right\|_1 \right\} = \begin{cases} 0 & \text{if } \|\overrightarrow{\alpha}\|_\infty \leq u, \\ -\infty & \text{otherwise.} \end{cases}$$

We would like the infimum to be bounded, so it is the first case that has to be considered. Recalling the expression of the infinity norm in Definition 2.30, this implies that $\alpha_i \geq -u$ and $\alpha_i \leq u$ for all i. Using this and all the above results in the general dual

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j t_i t_j K_{ij} - \sum_i \alpha_i q_i \\ \text{s.t.} \quad & \begin{cases} \max\{l,-u\} \le \alpha_i \le u, & \forall i, \\ s\sum_i \alpha_i t_i = 0, \\ z\sum_i \alpha_i = r. \end{cases} \end{aligned} \tag{5.66}$$

Observe that if $l = 0$ we recover (5.34), because it is assumed that $u > 0$. If $l \neq 0$ the only difference with respect to this problem is in the lower bound of the coefficients α_i. Depending on the specific value of l this bound is l itself or $-u$.

The l_2 case is very similar. Starting from the Lagrangian

$$\begin{aligned} \mathcal{L}\left(\overrightarrow{w}, b, \overrightarrow{\tau}, \overrightarrow{\xi}, \rho, \overrightarrow{\alpha}, \overrightarrow{\delta}\right) = \frac{1}{2}\|\overrightarrow{w}\|_2^2 + r\rho - l\sum_i \tau_i + \frac{u}{2}\|\overrightarrow{\xi}\|_2^2 \\ - \sum_i \alpha_i \left[t_i(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb) - q_i - \tau_i + \xi_i + z\rho\right] - \sum_i \delta_i\tau_i, \end{aligned} \tag{5.67}$$

the dual problem has the same form than (5.64). If we split the infimum in the different variables we arrive to

$$\begin{gathered} \inf_{\overrightarrow{w}}\left\{\tfrac{1}{2}\|\overrightarrow{w}\|_2^2 - \textstyle\sum_i \alpha_i t_i \langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle\right\} + \inf_b\left\{-sb\textstyle\sum_i \alpha_i t_i\right\} + \textstyle\sum_i \alpha_i q_i + \\ \inf_\rho\left\{\rho\left(r - z\textstyle\sum_i \alpha_i\right)\right\} + \inf_{\overrightarrow{\tau}}\left\{\textstyle\sum_i \tau_i\left(\alpha_i - \delta_i - l\right)\right\} + \inf_{\overrightarrow{\xi}}\left\{\tfrac{u}{2}\textstyle\sum_i \xi_i^2 - \textstyle\sum_i \alpha_i\xi_i\right\}. \end{gathered}$$

Since the only difference is in the infimum with respect to $\overrightarrow{\xi}$, the three first equalities of (5.33) still hold, as well as (5.65). This infimum is now differentiable, so there is no need to resort to the convex conjugate as before. Equaling to zero yields again (5.37), and substituting all this in the Lagrangian (5.67) gives the dual

$$\begin{aligned} \min_{\overrightarrow{\alpha}} \quad & \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j t_i t_j \left(K_{ij} + \frac{\delta_{ij}}{u}\right) - \sum_i \alpha_i q_i \\ \text{s.t.} \quad & \begin{cases} l \le \alpha_i, & \forall i, \\ s\sum_i \alpha_i t_i = 0, \\ z\sum_i \alpha_i = r, \end{cases} \end{aligned} \tag{5.68}$$

which is identical to (5.38), except for l now replacing the lower bound of 0.

LS–SVMs clearly belong to the l_2 case, as they penalize quadratically the slack variables. Instead of considering LS–SVC and LS–SVR separately, it suffices to study LS–SVR. In fact, having a look at (3.38) and (3.58), it is easy to see that LS–SVC is a particular case of LS–SVR where the slacks are changed to $\xi_i \leftarrow y_i \xi_i$ (this does not change the objective function, because the slacks are squared, and $y_i = \pm 1$ in classification).

In view of (5.68) and (3.61), we can transform the latter into the former by making $r = z = 0$ and $s = 1$. As for l, it should have the limit value $-\infty$, so that the coefficients α_i are unconstrained.

Before we can establish this result, it is useful to show an intermediate one:

Proposition 5.15. *Assume that $r = z = 0$, and $p = 2$ in (5.62). Taking the limit when $l \to -\infty$, the optimal solution of this problem has $\tau_i = 0 \; \forall i$.*

Proof. Let us assume for the time being that $l < 0$. Because of the other assumptions of the proposition, problem (5.30) becomes

$$\begin{aligned} \min_{\overrightarrow{w}, b, \overrightarrow{\tau}, \overrightarrow{\xi}} \quad & \frac{1}{2} \| \overrightarrow{w} \|_2^2 - l \sum_i \tau_i + \frac{u}{2} \| \overrightarrow{\xi} \|_2^2 \\ \text{s.t.} \quad & \begin{cases} t_i(\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle + sb) = q_i + \tau_i - \xi_i, & \forall i, \\ \tau_i \geq 0, & \forall i. \end{cases} \end{aligned}$$

Fixing temporarily $\overrightarrow{w}$ and b, and defining $h_i = t_i(\langle \overrightarrow{w}, \Phi(\overrightarrow{x}_i) \rangle + sb) - q_i$, the problem can be reformulated as

$$\begin{aligned} \min_{\overrightarrow{\tau}, \overrightarrow{\xi}} \quad & -l \sum_i \tau_i + \frac{u}{2} \| \overrightarrow{\xi} \|_2^2 \\ \text{s.t.} \quad & \begin{cases} h_i = \tau_i - \xi_i, & \forall i, \\ \tau_i \geq 0, & \forall i, \end{cases} \end{aligned}$$

which can obviously be minimized component–wise. Substituting $\tau_i = h_i + \xi_i$, we want to minimize for each i

$$-l(h_i + \xi_i) + \frac{u}{2} \xi_i^2.$$

Differentiating with respect to ξ_i and equaling to 0 yields

$$\xi_i = \frac{l}{u} < 0 \quad \Rightarrow \quad \tau_i = h_i + \frac{l}{u},$$

as $u > 0$ in (5.30). However, we must satisfy the constraint $\tau_i \geq 0$, so the above solution is only valid if $h_i \geq -l/u$. If this is not the case, we are compelled to choose $\xi_i \geq -h_i > l/u$. Since the above function of ξ_i is a parabola with its vertex in l/u, it is clear that we should pick a ξ_i as small as possible, which means taking $\xi_i = -h_i$ and $\tau_i = 0$.

To conclude the proof, we observe that in the limit $l \to -\infty$ it is always the case that $h_i < -l/u$, so that $\tau_i = 0 \; \forall i$. □

This proposition tells us that, with the assumptions made, in this limit we can remove $\overrightarrow{\tau}$ from the problem, as it is equal to $\overrightarrow{0}$. Thus, we can finally include LS–SVMs.

Proposition 5.16. *The primal* (5.30) *becomes the LS–SVR (and hence the LS–SVC) primal in the limit case* $l \to -\infty$*, by making* $p = 2$*,* $u = C$*,* $r = 0$*,* $s = 1$*,* $z = 0$*,* $t_i = +1, i = 1, \ldots, N$ *and* $q_i = y_i, i = 1, \ldots, N$*.*

Proof. By Proposition 5.15, it can be assumed that $\overrightarrow{\tau} = \overrightarrow{0}$, so we can remove it from the problem. Then we are left with minimizing

$$\frac{1}{2}\|\overrightarrow{w}\|_2^2 + \frac{C}{2}\|\overrightarrow{\xi}\|_2^2,$$

with respect to $\overrightarrow{w}$, b and $\overrightarrow{\xi}$, subject to constraints of the form

$$\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right)\rangle + b = y_i - \xi_i,$$

that is, the LS–SVR primal (3.58). □

As for the KKT conditions in this l_2 case, they are easily seen to be

$$
\begin{array}{ll}
\text{S} & \begin{cases} \overrightarrow{w}^* = \sum_i \alpha_i^* t_i \Phi\left(\overrightarrow{x}_i\right), \\ s \sum_i \alpha_i^* t_i = 0, \\ z \sum_i \alpha_i^* = r, \end{cases} \\
\text{DF} & \begin{cases} l \leq \alpha_i^*, \quad \forall i, \end{cases} \\
\text{PF} & \begin{cases} t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + s b^* \right) = q_i + \tau_i^* - \frac{\alpha_i^*}{u} - z\rho^*, & \forall i, \\ \tau_i^* \geq 0, & \forall i, \end{cases} \\
\text{CS} & \begin{cases} \left(\alpha_i^* - l\right) \tau_i^* = 0, \quad \forall i. \end{cases}
\end{array}
\tag{5.69}
$$

Recalling Definition 2.66, the KKT conditions are necessary and sufficient only if the objective function is differentiable, but this is not the case for $p = 1$ in (5.30). In this case, we can apply again that the optimal slacks are non–negative, so we can use the KKT conditions of a variant of (5.62) with constraints $\xi_i \geq 0 \; \forall i$, and where the slacks term in the objective function is $u \sum_i \xi_i$ instead of $u \sum_i |\xi_i|$. This problem has the same solution than (5.62) with $p = 1$. Its KKT conditions turn out to be

$$
\begin{array}{ll}
\text{S} & \begin{cases} \overrightarrow{w}^* = \sum_i \alpha_i^* t_i \Phi\left(\overrightarrow{x}_i\right), \\ s \sum_i \alpha_i^* t_i = 0, \\ z \sum_i \alpha_i^* = r, \end{cases} \\
\text{DF} & \begin{cases} \max\{l, -u\} \leq \alpha_i^* \leq u, \quad \forall i, \end{cases} \\
\text{PF} & \begin{cases} t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \rangle + s b^* \right) = q_i + \tau_i^* - \xi_i^* - z\rho^*, & \forall i, \\ \tau_i^* \geq 0, & \forall i, \\ \xi_i^* \geq 0, & \forall i, \end{cases} \\
\text{CS} & \begin{cases} \left(\alpha_i^* - l\right) \tau_i^* = 0, \quad \forall i, \\ \left(u - \alpha_i^*\right) \xi_i^* = 0, \quad \forall i. \end{cases}
\end{array}
\tag{5.70}
$$

The general SMO algorithm operates exactly in the same way than what was explained in §5.2.2. The only aspect that changes now is the lower bound in the coefficients α_i, so that instead of (5.51) we get

$$
\begin{array}{rcl}
\mathcal{I}_L & = & \{i : (t_i = +1 \wedge \alpha_i < v) \vee (t_i = -1 \wedge \alpha_i > \bar{l})\}, \\
\mathcal{I}_U & = & \{i : (t_i = +1 \wedge \alpha_i > \bar{l}) \vee (t_i = -1 \wedge \alpha_i < v)\}.
\end{array}
\tag{5.71}
$$

Here, $\bar{l}$ is equal to l in the l_2 formulations and to $\max\{l, -u\}$ in the l_1 ones, according to (5.68) and (5.66). As for the clips, we replace (5.52) by

$$\begin{array}{rcll} \overline{\lambda} & \leftarrow & \min\{\lambda, \alpha_L - \bar{l}\} & \text{if} \quad t_L = -1, \\ \overline{\lambda} & \leftarrow & \min\{\lambda, v - \alpha_L\} & \text{if} \quad t_L = +1, \\ \overline{\lambda} & \leftarrow & \min\{\lambda, v - \alpha_U\} & \text{if} \quad t_U = -1, \\ \overline{\lambda} & \leftarrow & \min\{\lambda, \alpha_U - \bar{l}\} & \text{if} \quad t_U = +1. \end{array} \tag{5.72}$$

It is also straightforward to see that the KKT conditions (5.70) and (5.69) can be summarized in the single equation $\Delta^* \leq 0$. There are again two modes of operation depending on z: if it is 0, the two points in the working set can belong to different classes and they directly give Δ, whereas if it is not 0, the two points must belong to the same class, and this class is the one that gives the value of Δ.

With these minor changes, LS–SVC and LS–SVR can also be included in Table 5.7: t_L need not be equal to t_U and $\mathcal{I}_L$ and $\mathcal{I}_U$ are the whole training set, as there are no restrictions in the search.

Next chapter deals with the convergence of Algorithm 9. We will also discuss how the proof of convergence can be adapted for the inclusion of LS–SVMs.

Chapter 6

Convergence of the General SMO Algorithm

The previous chapter dealt with the first main contribution of the thesis, arriving to a common formulation (primal and dual) that encompasses the problems we have seen in previous chapters. Applying the rationale of the SMO algorithm to this general dual yields a general SMO algorithm, which translates into the different variants of the method when we perform the substitutions of Table 5.6. These variants are summarized in Table 5.7.

This chapter describes the second main contribution of this dissertation, which is showing the convergence of this general SMO algorithm for solving the general dual (5.43), corresponding to the primal (5.30). Working with the general algorithm implies that with a single proof we can show the convergence of all the particular SMO variants. Moreover, the proof is much simpler and easy to follow than alternative proofs that have been proposed in the literature. These proofs will be discussed in the next section. The starting point of our proof is also different than those in other works, and extends the arguments given by Gilbert in [85] to show the convergence of his method.

As a consequence, the proof is also valid for the Gilbert, GSK and RCH–GSK methods, apart from the MDM and RCH–MDM algorithms, which are already included as particular cases of SMO for solving the MNP and NPP problems. Hence, we cover all the formulations and algorithms given in the thesis. As in the previous chapter, the extension for LS–SVMs is explained at the end of the chapter.

We begin with a revision of the literature on the topic of convergence of these methods. Having seen the different approaches, we explain ours. Next, we move to the proof itself, which is decomposed into a series of incremental lemmas and a final theorem. Last but

not least, we infer some additional consequences from the proof, such as the theoretical justification of the shrinking strategy adopted in the LIBSVM and SVM^{light} packages.

6.1 Literature on Convergence

The convergence of all these algorithms has not been given a lot of attention in recent years, except for the SMO method. The classical Gilbert's and MDM methods were originally suggested together with their proofs of asymptotic convergence in [85] and [28]. Their generalizations for the CH–NPP problem omitted the proofs of convergence. Anyhow, it is straightforward to generalize these proofs from having one convex hull in the MNP problem to two in the NPP one.

The same can be said about the RCH–GSK method, which was seen in §4.2.3.1 to be the application of the GSK method to the particular case of reduced convex hulls, where the selection of the extreme point with smallest margin is more complex. However, we described how the RCH–MDM method (Algorithm 8) is not the direct translation of the MDM algorithm for reduced convex hulls, so the classical proof in [28] fails in this case. Fortunately, we also saw that RCH–MDM is a particular case of SMO (Proposition 5.7), so we can apply the proofs of convergence of the latter to the former.

The original SMO algorithm was given in [26] with no proof of convergence. Recall that this version of the algorithm selected the pair of coefficients to be updated with two complex heuristics, so proving the convergence of the algorithm seemed quite a challenge. The suggestion of choosing the MVP with first–order selection in classification [59] and regression [69] simplified things, as it systematized the search of the coefficients with a fixed rule.

This allowed the first study on the subject for the classification case in [99]. Specifically, the authors showed that using first order selection and $\Delta < \epsilon$ as stopping criterion, with Δ given by (4.7) and building on the optimality condition (5.25), guarantees that the algorithm terminates in a finite number of iterations, no matter how small ϵ is. This proof was found to be incomplete and corrected in [100].

However, this is different from showing asymptotic convergence. In our general context, what we mean by "asymptotic convergence" is showing that any convergent subsequence $\left\{\overrightarrow{\alpha}^{t_j}\right\}$ (recall Definitions 2.3 and 2.5) has as its limit point an optimum, that is, a point $\overrightarrow{\alpha}^*$ such that $\Delta^* \leq 0$, according to (5.54), (5.59), (5.57) and (5.60). We have to deal with subsequences because the dual objective function in (5.43) can be convex but not strictly convex (recall Definition 2.60).

Looking at Figure 2.6, if the dual objective function were always strictly convex (this is the case when the matrix $\widetilde{\Omega}$ is positive definite), we would only have a unique minimum, so in fact the whole sequence $\left\{\overrightarrow{\alpha}^t\right\}$ would have this minimum as its limit point. Nevertheless, if the function is just convex, there may be more than one minimum, so it is possible that the sequence $\left\{\overrightarrow{\alpha}^t\right\}$ diverges because it oscillates between different minima. This is why we have to consider the subsequences of this sequence that converge each to a different minimum.

Showing that $\left\{\Delta^t\right\} \to 0$ is formally different from showing asymptotic convergence, because Δ in general is not a continuous function of $\overrightarrow{\alpha}$, as it is defined in (5.48) with a maximum and a minimum over sets which are not fixed, since they depend on the particular values $\overrightarrow{\alpha}^t$. Nevertheless, intuition tells us that these two facts are related, because every optimum must have a non–positive value of Δ, so if we decrease it to 0 we should be getting closer to an optimum.

Some proofs of convergence were then given for formulations somewhat different from the SVC problem, such as the one in [101], or for selections of the working set different from first order, as in [102]. A cornerstone was the paper by Chih-Jen Lin that showed the asymptotic convergence of the SVM^{light} algorithm [71]. Recall that the SMO algorithm with first order selection was a particular case of SVM^{light} where the working set size is restricted to 2, so it is essentially included in this proof.

However, the proof is quite involved, as it is general so as to cover SVM^{light} with any working set size. One of the drawbacks of being so general is that an assumption is needed so that any submatrix of the kernel matrix, taken from the indices of the working set, is positive definite. This happens for instance when the Gaussian kernel is being used and all points are distinct (see Theorem 2.38), but whenever we have two identical points the assumption does not hold.

Lin showed later that for the particular case of SMO this assumption is not needed [103]. This is useful, but does not simplify the proof of [71]. Building on this last work, he also justified the shrinking and caching strategies of SVM^{light} in [104], as well as seeing that for this algorithm we also have $\left\{\Delta^t\right\} \to 0$.

Even so, still it was only SVC and first order selection that were being considered. Concentrating on SMO instead of on SVM^{light}, [29] is the most complete reference regarding convergence. Here other selections are studied, including second order, and other formulations are considered in an appendix, such as SVR, OC and ν–SVC, but in a slightly different way. Another recent work concentrating on the convergence of SMO for SVR is [105], but it builds on the SMO version of [70], which we know from §4.1.1.1 that is not the same as ours.

The proof we give in this chapter is valid for both first and second order selections, as well as some other variants studied by Lin's team, and also covers more formulations than just SVC. In fact, we cover not only SVR, OC and ν–SVC as in [29], but also the MNP and NPP problems. On the other hand, it differs in its approach: while the proof in [29] uses a complex counting argument to arrive to a contradiction, here we generalize the arguments given by Gilbert in its proof of convergence for his algorithm.

As a result, this method and its extensions GSK and RCH–GSK are also covered by the proof (only a minor change is needed), which is not the case for Lin's proof. What is more, the final result is far simpler and intuitive than the reasonings by Lin. Unfortunately, a drawback is that our simplified reasoning is not valid for SVM^{light}, being only applicable to SMO/MDM and GSK. Despite this, it is of interest to have a much simplified proof for SMO, as it is still the state–of–the–art algorithm in solving SVM and geometric tasks.

Our proof consists of three basic steps, namely:

1. To bound the distance $\left\| \overrightarrow{w}^t - \overrightarrow{w}^* \right\|_2$ between the iterates $\overrightarrow{w}^t = \sum_i \alpha_i^t t_i \widetilde{\Phi}\left(\overrightarrow{x}_i\right)$ and the optimal $\overrightarrow{w}^*$ in the form $(1/2)\left\| \overrightarrow{w}^t - \overrightarrow{w}^* \right\|_2^2 \leq \mathcal{D}^t - \mathcal{D}^*$.
2. To prove that some subsequence $\left\{\overrightarrow{\alpha}^{t_j}\right\}$ converges to an optimal $\overrightarrow{\alpha}^*$.
3. To conclude that $\left\{\overrightarrow{w}^t\right\} \rightarrow \overrightarrow{w}^*$.

Observe that the final goal is to show that we are converging to the optimal weight vector in the primal. In fact, this is what we really want, for in the end we are looking for the optimal hyperplane (the bias b^* is determined from the KKT conditions, once we have -approximately- converged). This is different to the approach of Lin and his team; they concentrate on the sequence $\left\{\overrightarrow{\alpha}^t\right\}$ and ignore the evolution of the weight vector.

We find more sensible to concentrate on the sequence $\left\{\overrightarrow{w}^t\right\}$, because it is easily shown that the optimal weight vector $\overrightarrow{w}^*$ is unique, no matter whether the dual function is convex or strictly convex. Thus, this sequence converges globally independently of the sample, contrary to the sequence $\left\{\overrightarrow{\alpha}^t\right\}$, which has to be decomposed in [29] in its convergent subsequences. Note that in our second step we just need to prove the convergence to an optimum of one of these subsequences, instead of proving the convergence of all of them.

6.2 Proof of Convergence

In the sequel we consider the primal formulation (5.30), whose dual can be written in the form (5.43). This is the problem addressed by Algorithm 9, and splits in two slightly different problems when $p = 1$ (Equation (5.34)) and $p = 2$ (Equation (5.38)).

Specifically, we consider the case in Algorithm 9 where t_U need not be equal to t_L, and the pair selection directly gives Δ. Recall from the discussion in §5.2.2 that if $t_U = t_L$, we have to select separately in each class, and define Δ as the maximum between Δ_+ and Δ_-. With Δ defined in this way, the proof is analogous, but we need to consider separately the iterations for which we select $\Delta = \Delta_+$ and those for which $\Delta = \Delta_-$. Hence, we stick to the standard case for the sake of simplicity.

Throughout this section we briefly mention how the proof would have to be completed for the case where $t_L = t_U$. The extension for the more general primal (5.62) is postponed till §6.4.

6.2.1 Preliminary Observations

In this section some preliminary remarks are given before starting the proof itself. Observe that in the first step we are not only assuming that $\overrightarrow{w}^*$ is unique, but also that the value $\mathcal{D}^*$ is unique. However, this last fact is obvious because the dual is convex, so all optima are global, and the dual optimal value $\mathcal{D}^*$ is thus unique (see again Figure 2.6).

Regarding the uniqueness of $\overrightarrow{w}^*$, in the l_1 case this means that $\overrightarrow{w}^*$ is unique, because $\overrightarrow{w} = \overrightarrow{\widetilde{w}}$. However, in the l_2 case, this means that both $\overrightarrow{w}^*$ and $\overrightarrow{\xi}^*$ are unique, in view of (5.39).

The general primal formulation (5.30) is a convex problem with affine inequality constraints, so strong duality holds by Theorem 2.63. Remember as well that we are assuming that the values used in (5.30) ensure that there is at least an optimal finite solution.

Therefore, we will have $\mathcal{P}^* = -\mathcal{D}^*$, where $\mathcal{P}^*$ is the optimal value of the primal objective function for any optimal solution $\left(\overrightarrow{w}^*, b^*, \overrightarrow{\xi}^*, \rho^*\right)$. This value is also unique, because the function $\mathcal{P}$ is convex too. The minus sign comes from the fact that we are solving a minimization in the dual instead of the maximization indicated in Definition 2.61.

To see that $\overrightarrow{\widetilde{w}}^*$ is unique, let us consider the augmented vector $\overrightarrow{\pi} = \left(\overrightarrow{w}^*, b^*, \overrightarrow{\xi}^*, \rho^*\right)$. The primal (5.30) is still convex in terms of $\overrightarrow{\pi}$, so the set of optimal solutions $\overrightarrow{\pi}^*$ should be convex as well by Theorem 2.50.

Denoting $\overrightarrow{\pi}_1^* = \left(\overrightarrow{w}_1^*, b_1^*, \overrightarrow{\xi}_1^*, \rho_1^*\right)$ and $\overrightarrow{\pi}_2^* = \left(\overrightarrow{w}_2^*, b_2^*, \overrightarrow{\xi}_2^*, \rho_2^*\right)$ as two different hypothetical solutions in this set, the parametrized solution $\overrightarrow{\pi}_\delta^* = (1-\delta)\overrightarrow{\pi}_1^* + \delta\overrightarrow{\pi}_2^*$ should by convexity (cf. Definition 2.40) be an optimal solution as well for every $0 \leq \delta \leq 1$. Thus, we would have $\mathcal{P}(\overrightarrow{\pi}_1^*) = \mathcal{P}(\overrightarrow{\pi}_2^*) = \mathcal{P}(\overrightarrow{\pi}_\delta^*)$.

We can now establish the following result:

Proposition 6.1. *All optimal solutions* $\left(\overrightarrow{w}^*, b^*, \overrightarrow{\xi}^*, \rho^*\right)$ *for the general primal* (5.30) *share the same* $\overrightarrow{w}^*$*, which is unique. Moreover, for the* l_2 *formulations* $\overrightarrow{\xi}^*$ *is also unique.*

Proof. We have to distinguish between the two cases $p = 2$ and $p = 1$. If $p = 2$ and we build the convex function $\psi(\delta) = \mathcal{P}(\overrightarrow{\pi}_\delta^*) - \mathcal{P}(\overrightarrow{\pi}_1^*)$, we obtain

$$\begin{aligned}\psi(\delta) &= \frac{\delta}{2}\left[(\delta-2)\left\|\overrightarrow{w}_1^*\right\|_2^2 + 2(1-\delta)\left\langle\overrightarrow{w}_1^*, \overrightarrow{w}_2^*\right\rangle + \delta\left\|\overrightarrow{w}_2^*\right\|_2^2\right] + r\delta\left(\rho_2^* - \rho_1^*\right) + \\ &\quad \frac{u\delta}{2}\left[(\delta-2)\left\|\overrightarrow{\xi}_1^*\right\|_2^2 + 2(1-\delta)\left\langle\overrightarrow{\xi}_1^*, \overrightarrow{\xi}_2^*\right\rangle + \delta\left\|\overrightarrow{\xi}_2^*\right\|_2^2\right].\end{aligned}$$

Differentiating twice with respect to δ and using the fact that $\psi(\delta) = 0$ for $0 \leq \delta \leq 1$, we should get also a null derivative in that interval, which means

$$\left\|\overrightarrow{w}_1^*\right\|_2^2 - 2\left\langle\overrightarrow{w}_1^*, \overrightarrow{w}_2^*\right\rangle + \left\|\overrightarrow{w}_2^*\right\|_2^2 + \left\|\overrightarrow{\xi}_1^*\right\|_2^2 - 2\left\langle\overrightarrow{\xi}_1^*, \overrightarrow{\xi}_2^*\right\rangle + \left\|\overrightarrow{\xi}_2^*\right\|_2^2 = 0,$$

so that $\overrightarrow{w}_1^* = \overrightarrow{w}_2^*$ and $\overrightarrow{\xi}_1^* = \overrightarrow{\xi}_2^*$.

Operating in the same way for $p = 1$ we get

$$\begin{aligned}\psi(\delta) &= \frac{\delta}{2}\left[(\delta-2)\left\|\overrightarrow{w}_1^*\right\|_2^2 + 2(1-\delta)\left\langle\overrightarrow{w}_1^*, \overrightarrow{w}_2^*\right\rangle + \delta\left\|\overrightarrow{w}_2^*\right\|_2^2\right] + r\delta\left(\rho_2^* - \rho_1^*\right) + \\ &\quad u\delta\sum_i\left(\left(\overrightarrow{\xi}_2^*\right)_i - \left(\overrightarrow{\xi}_1^*\right)_i\right),\end{aligned}$$

and differentiating twice now means that

$$\left\|\overrightarrow{w}_1^*\right\|_2^2 - 2\left\langle\overrightarrow{w}_1^*, \overrightarrow{w}_2^*\right\rangle + \left\|\overrightarrow{w}_2^*\right\|_2^2 = 0,$$

so $\overrightarrow{w}_1^* = \overrightarrow{w}_2^*$. □

This shows that $\overrightarrow{\widetilde{w}}^*$ is unique in both cases. Moreover, since $\overrightarrow{\xi}^* = \overrightarrow{\alpha}^*/u$, it follows that $\overrightarrow{\alpha}^*$ is also unique when $p = 2$. Alternatively, we can arrive to this same result by noting that $\widetilde{\Omega}$ in (5.43) is always positive definite (hence the dual is strictly convex), since the kernel matrix is positive semidefinite and we are adding $1/u$ to its main diagonal.

In the l_1 case we only get that $\overrightarrow{w}^*$ is unique, but $\overrightarrow{\xi}^*$ need not be so. Nor does $\overrightarrow{\alpha}^*$, because $\widetilde{\Omega}$ is positive semidefinite, but no term is added to its main diagonal. This case was studied for the l_1–SVC formulation in [106].

It is also convenient to check that the iterates given by the general SMO algorithm while solving dual (5.43) are bounded from below, whenever there exists an optimal solution:

Proposition 6.2. *If an optimal solution exists, the iterates* $\{\mathcal{D}^t\}$ *given by Algorithm 9 are bounded from below for all formulations considered.*

Proof. In view of (5.43), the dual can be written as $\mathcal{D}^t = (1/2)\left\|\overrightarrow{\widetilde{w}}^t\right\|_2^2 - \langle \overrightarrow{\alpha}^t, \overrightarrow{q} \rangle$. Because the first term is a norm, $\mathcal{D}^t \geq -\langle \overrightarrow{\alpha}^t, \overrightarrow{q} \rangle$. Hence, we would like to bound from above the quantity $\langle \overrightarrow{\alpha}^t, \overrightarrow{q} \rangle$.

In the particular cases where $\overrightarrow{q} = \overrightarrow{0}$ (ν–SVC, OC/MNP and (R)CH–Margin/NPP in Table 5.6), this is trivial and we have $\mathcal{D}^t \geq 0$ for every t.

For the rest of cases, by Cauchy's inequality, $\langle \overrightarrow{\alpha}^t, \overrightarrow{q} \rangle \leq \|\overrightarrow{q}\|_2 \left\|\overrightarrow{\alpha}^t\right\|_2$. According to Table 5.6, $\|\overrightarrow{q}\|_2$ is a finite value, because so are its components q_i (it is assumed that no label y_i is infinite, not even in regression).

Consequently, for the l_1 formulations the result is trivial, since $\left\|\overrightarrow{\alpha}^t\right\|_2 = \sqrt{\sum_i (\alpha_i^t)^2} \leq u\sqrt{|\mathcal{I}|}$, where $|\mathcal{I}|$ denotes the dimension of the vector $\overrightarrow{\alpha}^t$, which is $2N$ for regression and N for the rest of formulations.

At this point, we are left with the l_2 formulations with $\overrightarrow{q} \neq \overrightarrow{0}$. Here $\overrightarrow{\alpha}^t$ is not bounded from above by $\overrightarrow{u}$, so an alternative reasoning is needed.

If the constraint $z\sum_i \alpha_i = r$ is non–trivial, we have $r/z = \sum_i \alpha_i^t = \sum_i |\alpha_i^t| = \left\|\overrightarrow{\alpha}^t\right\|_1$, because $\alpha_i^t \geq 0$ for those formulations. Using the general fact that

$$\|\overrightarrow{\alpha}\|_2 \leq \|\overrightarrow{\alpha}\|_1 \leq \sqrt{|\mathcal{I}|}\,\|\overrightarrow{\alpha}\|_2$$

it follows that $\left\|\overrightarrow{\alpha}^t\right\|_2 \leq r/z$, so the sequence $\{\mathcal{D}^t\}$ is also bounded from below.

Finally, note that when the constraint $z\sum_i \alpha_i = r$ is trivial, $\overrightarrow{\alpha}^0 = \overrightarrow{0}$ is a feasible solution. We can also write $\left\|\overrightarrow{\widetilde{w}}^t\right\|_2^2 = \left\langle \overrightarrow{\alpha}^t, \widetilde{\Omega}\overrightarrow{\alpha}^t \right\rangle$ and use the general result that

$$\lambda_{\min}\left(\widetilde{\Omega}\right)\langle\overrightarrow{\alpha},\overrightarrow{\alpha}\rangle \leq \left\langle\overrightarrow{\alpha},\widetilde{\Omega}\overrightarrow{\alpha}\right\rangle,$$

where $\lambda_{\min}\left(\widetilde{\Omega}\right)$ stands for the minimal eigenvalue of matrix $\widetilde{\Omega}$. This eigenvalue is positive, because $\widetilde{\Omega}$ is positive definite for this case. It follows that $\|\overrightarrow{q}\|_2 \left\|\overrightarrow{\alpha}^t\right\|_2 = \|\overrightarrow{q}\|_2 \left\|\overrightarrow{\widetilde{w}}^t\right\|_2 / \sqrt{\lambda_{\min}(\widetilde{Q})}$.

Starting the algorithm in $\overrightarrow{\alpha}^0 = \overrightarrow{0}$, we get for every t that

$$\frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^t\right\|_2^2 \leq \langle\overrightarrow{\alpha}^t,\overrightarrow{q}\rangle \leq \|\overrightarrow{q}\|_2 \left\|\overrightarrow{\alpha}^t\right\|_2 \leq \frac{\|\overrightarrow{q}\|_2 \left\|\overrightarrow{\widetilde{w}}^t\right\|_2}{\sqrt{\lambda_{\min}\left(\widetilde{Q}\right)}},$$

so $\left\|\overrightarrow{\widetilde{w}}^t\right\|_2 \leq 2\|\overrightarrow{q}\|_2 / \sqrt{\lambda_{\min}(\widetilde{Q})}$, and therefore $\langle\overrightarrow{\alpha}^t,\overrightarrow{q}\rangle \leq 2\|\overrightarrow{q}\|_2^2 / \lambda_{\min}(\widetilde{Q})$, so the result follows. □

In the sequel, we will distinguish between clipped and non–clipped updates. By "non–clipped updates" we mean those updates where the value of $\overline{\lambda}$ in (5.46) can be safely used. Conversely, "clipped updates" refer to those updates where this value has to be clipped with (5.52). We proceed now to show that the sequence $\left\{\mathcal{D}^t\right\}$ is strictly monotonically decreasing (cf. Definition 2.2):

Proposition 6.3. *The sequence $\left\{\mathcal{D}^t\right\}$ given by Algorithm 9 is strictly monotonically decreasing.*

Proof. Let us choose any particular t. If a non–clipped update can be performed, we have by virtue of (5.49) that

$$\mathcal{D}^t - \mathcal{D}^{t+1} = \frac{\left(\Delta^t\right)^2}{2\left\|\overrightarrow{\widetilde{z}}_{U^tL^t}\right\|_2^2} \geq \frac{\left(\Delta^t\right)^2}{2D^2},$$

where $D = \max_{p,q}\left\|\widetilde{\Phi}\left(\overrightarrow{x}_p\right) - \widetilde{\Phi}\left(\overrightarrow{x}_q\right)\right\|_2$, which is obviously a finite value because the number of samples is finite.

On the other hand, if we need to perform a clipped update by using (5.52), we have four possible clips. Let us examine the first one. If this is applied, it is because $\alpha_{L^t}^t \leq \Delta^t / \left\|\overrightarrow{\widetilde{z}}_{U^tL^t}\right\|_2^2$, which means that $\alpha_{L^t}^t \Delta^t \geq \left(\alpha_{L^t}^t\right)^2 \left\|\overrightarrow{\widetilde{z}}_{U^tL^t}\right\|_2^2$. (5.52) makes that $\lambda_{L^t}^t = t_{L^t}\alpha_{L^t}^t$. In view of (5.45) and using $t_i = \pm 1$, we also have

$$\mathcal{D}^t - \mathcal{D}^{t+1} = \lambda_{L^t}^t t_{L^t} \Delta^t - \frac{\left(\lambda_{L^t}^t\right)^2}{2} \left\| \overrightarrow{\widetilde{z}}_{UL} \right\|_2^2 = \alpha_{L^t}^t \Delta^t - \frac{\left(\alpha_{L^t}^t\right)^2}{2} \left\| \overrightarrow{\widetilde{z}}_{U^t L^t} \right\|_2^2,$$

so it follows that $\mathcal{D}^t - \mathcal{D}^{t+1} \geq \alpha_{L^t}^t \Delta^t / 2$.

A similar reasoning for the second clip yields that $\mathcal{D}^t - \mathcal{D}^{t+1} \geq (v - \alpha_{L^t}^t)\Delta^t/2$, whereas the two remaining clips are treated in the same way, but considering $\lambda_{U^t}^t$ instead of $\lambda_{L^t}^t$, and using $\lambda_{U^t} = -t_{L^t} t_{U^t} \lambda_{L^t}$. □

After considering all the possible cases in the above proposition, we can bound the dual gain as follows:

$$\mathcal{D}^t - \mathcal{D}^{t+1} \quad \geq \quad \min \left\{ \frac{(\Delta^t)^2}{2D^2}, \frac{\alpha_{L^t}^t \Delta^t}{2}, \frac{\alpha_{U^t}^t \Delta^t}{2}, \frac{\left(u - \alpha_{L^t}^t\right) \Delta^t}{2}, \frac{\left(u - \alpha_{U^t}^t\right) \Delta^t}{2} \right\}. \quad (6.1)$$

Of course, this formula is valid for the l_1 cases, where $v = u$. For the l_2 formulations there are no clips with u. It is also worth noting that we can have a very similar bound if we are using second order selection. (5.61) tells us that

$$\frac{\left(\Delta_2^t\right)^2}{2 \left\| \widetilde{\Phi}\left(\overrightarrow{x}_{L_2^t}\right) - \widetilde{\Phi}\left(\overrightarrow{x}_{U_2^t}\right) \right\|_2^2} \geq \frac{\left(\Delta^t\right)^2}{2 \left\| \widetilde{\Phi}\left(\overrightarrow{x}_{L^t}\right) - \widetilde{\Phi}\left(\overrightarrow{x}_{U^t}\right) \right\|_2^2} \geq \frac{\left(\Delta^t\right)^2}{2D^2},$$

because the MVP is also considered in the search. Here we use a specific subscript of 2 on Δ for second order. The same notation is used next; we can write the bound

$$\mathcal{D}^t - \mathcal{D}^{t+1} \quad \geq \quad \min \left\{ \frac{(\Delta^t)^2}{2D^2}, \frac{\alpha_{L_2^t}^t \Delta_2^t}{2}, \frac{\alpha_{U_2^t}^t \Delta_2^t}{2}, \frac{\left(u - \alpha_{L_2^t}^t\right) \Delta_2^t}{2}, \frac{\left(u - \alpha_{U_2^t}^t\right) \Delta_2^t}{2} \right\}, \quad (6.2)$$

since (5.52) is independent of the pair selection.

At this point we are ready to prove the first two steps.

6.2.2 Step 1

Our goal in this step is to show the following result.

Lemma 6.4. *In Algorithm 9, for any iterate* $\overrightarrow{\widetilde{w}}^t = \sum \alpha_i^t t_i \widetilde{\Phi}(\overrightarrow{x}_i)$ *it holds that* $(1/2)\|\overrightarrow{\widetilde{w}}^t - \overrightarrow{\widetilde{w}}^*\|_2^2 \leq \mathcal{D}^t - \mathcal{D}^*$.

Proof. Recall that by strong duality it must hold that $\mathcal{P}^* = -\mathcal{D}^*$, where $\mathcal{P}^*$ is the primal value for any optimal solution $\left(\overrightarrow{w}^*, b^*, \overrightarrow{\xi}^*, \rho^*\right)$ and $\mathcal{D}^*$ is the dual value for any optimal solution $\overrightarrow{\alpha}^*$. This, together with the uniqueness of $\overrightarrow{\widetilde{w}}^*$ by Proposition 6.1, implies that $\overrightarrow{\widetilde{w}}^*$ is both primal and dual feasible.

We distinguish again between the cases with $p = 1$ and $p = 2$. If $p = 1$, $\overrightarrow{\widetilde{w}}^* = \overrightarrow{w}^*$ and $\widetilde{\Phi}(\cdot) = \Phi(\cdot)$. Using (5.30), for any dual feasible $\overrightarrow{\widetilde{w}}^t = \sum \alpha_i^t t_i \widetilde{\Phi}(\overrightarrow{x}_i)$ we can write

$$\begin{aligned}\left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle &= \sum_i \alpha_i^t t_i \left\langle \overrightarrow{\widetilde{w}}^*, \widetilde{\Phi}(\overrightarrow{x}_i) \right\rangle = \sum_i \alpha_i^t t_i \left(\langle \overrightarrow{w}^*, \Phi(\overrightarrow{x}_i) \rangle + s b^* \right) \\ &\geq \sum_i \alpha_i^t \left(q_i - \xi_i^* - z\rho^* \right),\end{aligned}$$

as $s \sum_i \alpha_i^t t_i = 0$. Hence, it is easy to derive

$$\begin{aligned}\frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^t - \overrightarrow{\widetilde{w}}^* \right\|_2^2 &= \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^t \right\|_2^2 - \left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle + \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^* \right\|_2^2 \\ &\leq \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^t \right\|_2^2 - \sum_i \alpha_i^t \left(q_i - \xi_i^* - z\rho^* \right) + \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^* \right\|_2^2 \\ &= \mathcal{D}^t - \sum_i \alpha_i^t \left(-\xi_i^* - z\rho^* \right) + \frac{1}{2}\| \overrightarrow{w}^* \|_2^2 \\ &= \mathcal{D}^t - \sum_i \alpha_i^t \left(-\xi_i^* \right) + r\rho^* + \frac{1}{2}\| \overrightarrow{w}^* \|_2^2 \\ &\leq \mathcal{D}^t + u \sum_i \xi_i^* + r\rho^* + \frac{1}{2}\| \overrightarrow{w}^* \|_2^2 \\ &= \mathcal{D}^t + \mathcal{P}^* \\ &= \mathcal{D}^t - \mathcal{D}^*,\end{aligned}$$

where we used (5.34) and (5.30).

If $p = 2$, $\overrightarrow{\widetilde{w}}^*$ is given by (5.39) and $\widetilde{\Phi}(\cdot)$ by (5.40), so that $t_i \left(\left\langle \overrightarrow{\widetilde{w}}^*, \widetilde{\Phi}(\overrightarrow{x}_i) \right\rangle + s b^* \right) = t_i \left(\langle \overrightarrow{w}^*, \Phi(\overrightarrow{x}_i) \rangle + s b^* \right) + \xi_i^*$.

Operating similarly to above,

$$\begin{aligned}\left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle &= \sum_i \alpha_i^t t_i \left\langle \overrightarrow{\widetilde{w}}^*, \widetilde{\Phi}\left(\overrightarrow{x}_i\right) \right\rangle = \sum_i \alpha_i^t t_i \left(\left\langle \overrightarrow{\widetilde{w}}^*, \widetilde{\Phi}\left(\overrightarrow{x}_i\right) \right\rangle + sb^* \right) \\ &= \sum_i \alpha_i^t \left[t_i \left(\langle \overrightarrow{w}, \Phi\left(\overrightarrow{x}_i\right) \rangle + sb \right) + \xi_i^* \right] \geq \sum_i \alpha_i^t \left(q_i - z\rho^* \right).\end{aligned}$$

Now it follows that

$$\begin{aligned}\frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^t - \overrightarrow{\widetilde{w}}^* \right\|_2^2 &= \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^t \right\|_2^2 - \left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle + \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^* \right\|_2^2 \\ &\leq \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^t \right\|_2^2 - \sum_i \alpha_i^t \left(q_i - z\rho^* \right) + \frac{1}{2}\left\| \overrightarrow{\widetilde{w}}^* \right\|_2^2 \\ &= \mathcal{D}^t - \sum_i \alpha_i^t \left(-z\rho^* \right) + \frac{1}{2}\left\| \overrightarrow{w}^* \right\|_2^2 + \frac{1}{2u}\left\| \overrightarrow{\alpha}^* \right\|_2^2 \\ &= \mathcal{D}^t + r\rho^* + \frac{1}{2}\left\| \overrightarrow{w}^* \right\|_2^2 + \frac{1}{2u}\left\| \overrightarrow{\alpha}^* \right\|_2^2 \\ &= \mathcal{D}^t + r\rho^* + \frac{1}{2}\left\| \overrightarrow{w}^* \right\|_2^2 + \frac{u}{2}\left\| \overrightarrow{\xi}^* \right\|_2^2 \\ &= \mathcal{D}^t + \mathcal{P}^* \\ &= \mathcal{D}^t - \mathcal{D}^*,\end{aligned}$$

where we used (5.38), (5.30) and (5.37). □

6.2.3 Step 2

Here our goal is to find a subsequence $\left\{\overrightarrow{\alpha}^{t_j}\right\}$ that converges to an optimal point. To do so, the strategy we follow is to find first a subsequence $\left\{\Delta^{t_j}\right\}$ that converges to 0. We begin by proving the following.

Proposition 6.5. *There is a subsequence $\{t_j\}$ of iterations of Algorithm 9 where clipping does not take place.*

Proof. Assume to the contrary that there is an iteration T after which we perform clipped updates for all $t \geq T$. By virtue of (5.52), clipping means that at least one of the coefficients α_{L^t} or α_{U^t} ends in either the 0 or u bounds (if $p = 2$, only in 0).

Let us denote as N_B^t the number of coefficients that are in bounds at iteration t. Since we are updating only two coefficients, this number is non–decreasing for $t \geq T$ and, hence, stabilizes because $N_B^t \leq N$ and N is finite. Once N_B^t stabilizes, if clipped updates

take place, one multiplier must arrive to a bound and the other one leave a bound. Considering the eight possible cases we might encounter:

1. $(t_{L^t} = -1, t_{U^t} = +1)$: $\alpha_{L^t}^t$ becomes 0, and $\alpha_{U^t}^t$ changes from u to $u - \alpha_{L^t}^t$,
2. $(t_{L^t} = -1, t_{U^t} = -1)$: $\alpha_{L^t}^t$ becomes 0, and $\alpha_{U^t}^t$ changes from 0 to $\alpha_{L^t}^t$,
3. $(t_{L^t} = +1, t_{U^t} = +1)$: $\alpha_{L^t}^t$ changes from 0 to $\alpha_{U^t}^t$, and $\alpha_{U^t}^t$ becomes 0,
4. $(t_{L^t} = -1, t_{U^t} = +1)$: $\alpha_{L^t}^t$ changes from u to $u - \alpha_{U^t}^t$, and $\alpha_{U^t}^t$ becomes 0,
5. $(t_{L^t} = +1, t_{U^t} = +1)$: $\alpha_{L^t}^t$ becomes u, and $\alpha_{U^t}^t$ changes from u to $\alpha_{L^t}^t$,
6. $(t_{L^t} = +1, t_{U^t} = -1)$: $\alpha_{L^t}^t$ becomes u, and $\alpha_{U^t}^t$ changes from 0 to $u - \alpha_{L^t}^t$,
7. $(t_{L^t} = +1, t_{U^t} = -1)$: $\alpha_{L^t}^t$ changes from 0 to $u - \alpha_{U^t}^t$, and $\alpha_{U^t}^t$ becomes u,
8. $(t_{L^t} = -1, t_{U^t} = -1)$: $\alpha_{L^t}^t$ changes from u to $\alpha_{U^t}^t$, and $\alpha_{U^t}^t$ becomes u,

(which are reduced to cases 2, 3, 5, and 8 if $t_L = t_U$ in Table 5.7, and to cases 2 and 3 if $p = 2$), it is clear that in each such iteration t, the multipliers α_i^{t+1} are the result of shuffling around the values in the set $V^t = \{\overrightarrow{0}, \overrightarrow{u}, \overrightarrow{\alpha}^t, \overrightarrow{u} - \overrightarrow{\alpha}^t\}$ (analogously, this set is reduced to $V^t = \{\overrightarrow{0}, \overrightarrow{\alpha}^t\}$ if $p = 2$). Observe that V^t must coincide with V^T, as only clipped iterations happen for $t \geq T$.

However, since there is a finite number of such shuffles, we must arrive at a pair t_1, t_2 with $t_2 > t_1$, $\overrightarrow{\alpha}^{t_1} = \overrightarrow{\alpha}^{t_2}$ and, thus, $\mathcal{D}^{t_1} = \mathcal{D}^{t_2}$, which is not possible because the dual decreases strictly by Proposition 6.3. Hence, the result follows.

The above subsequence $\{t_j\}$ is formed by taking the iterations where we do not apply clipping with (5.52), so that in fact we apply (5.46) for every t_j. □

The desired subsequence $\left\{\Delta^{t_j}\right\}$ is the one taken from these iterations, because of the following immediate result.

Proposition 6.6. *For the subsequence $\{t_j\}$ just obtained, $\left\{\Delta^{t_j}\right\} \to 0$.*

Proof. Since no clipping takes place at the $\{t_j\}$ iterations, we update with (5.46) and the first bound in (6.1) (or in (6.2) if we use second order) is the one that holds, so

$$\mathcal{D}^{t_j} - \mathcal{D}^{t_j+1} \geq \frac{\left(\Delta^{t_j}\right)^2}{2D^2},$$

and we have $\{\Delta^{t_j}\} \to 0$ since so does $\{\mathcal{D}^{t_j} - \mathcal{D}^{t_{j+1}}\}$. This last result holds because the whole sequence $\{\mathcal{D}^t\}$ is strictly monotonically decreasing by Proposition 6.3 and bounded by Proposition 6.2, so it is convergent by Theorem 2.17. □

We complete next Step 2 by proving the following.

Lemma 6.7. *There is a subsequence* $\{t_j\}$ *in Algorithm 9 such that* $\{\overrightarrow{\alpha}^{t_j}\} \to \overrightarrow{\alpha}^*$*, with* $\overrightarrow{\alpha}^*$ *an optimal point.*

Proof. Let $\{t_j\}$ the subsequence of Proposition 6.6. Due to Proposition 6.2, the sequence $\{\mathcal{D}^t\}$ is bounded from below, so the set $\{\overrightarrow{\alpha}^t : \mathcal{D}^t \leq \mathcal{D}^0\}$ is compact (Algorithm 9 is initialized from a finite feasible solution $\overrightarrow{\alpha}^0$, which always exists if (5.30) is feasible), and the sequence $\{\overrightarrow{\alpha}^{t_j}\}$ lies in this set.

By Definition 2.34, there must be a subsequence of $\{\overrightarrow{\alpha}^{t_j}\}$ that converges to a point in this set. Let us call this limit point $\overrightarrow{\alpha}^*$, and the convergent subsequence $\{\overrightarrow{\alpha}^{t_{j_k}}\}$. Clearly, $\overrightarrow{\alpha}^*$ is dual feasible. For it to be optimal, we have to prove that $\Delta^* \leq 0$, according to (5.59) and (5.54).

Assume to the contrary that $\Delta^* > 0$. The fact that $\{\overrightarrow{\alpha}^{t_{j_k}}\} \to \overrightarrow{\alpha}^*$ and the gradient's continuity imply that $\forall \epsilon > 0$ there will be a T_1 large enough so that $|t_i \nabla \mathcal{D}_i^{t_{j_k}} - t_i \nabla \mathcal{D}_i^*| < \epsilon \ \forall i$ and for every $t_{j_k} > T_1$.

Moreover, convergence also implies that there must be a T_2 from which every candidate for L and U in $\overrightarrow{\alpha}^*$ is also a candidate for L and U in every $\overrightarrow{\alpha}^{t_{j_k}}$ with $t_{j_k} > T_2$ (this was shown in [78] for l_1–SVC, but it can be easily generalized to all formulations using Table 5.7).

Specifically, for the MVP (L^*, U^*) at $\overrightarrow{\alpha}^*$, we have that L^* will always be a candidate for $L^{t_{j_k}}$ and U^* will be so for $U^{t_{j_k}}$. Hence, for any $t_{j_k} > \max\{T_1, T_2\}$ we get

$$\begin{aligned}
\Delta^{t_{j_k}} &= \max_{\mathcal{I}_U^{t_{j_k}}} \{t_i \nabla \mathcal{D}_i^{t_{j_k}}\} - \min_{\mathcal{I}_L^{t_{j_k}}} \{t_i \nabla \mathcal{D}_i^{t_{j_k}}\} \\
&\geq t_{U^*} \nabla \mathcal{D}_{U^*}^{t_{j_k}} - t_{L^*} \nabla \mathcal{D}_{L^*}^{t_{j_k}} \\
&\geq (t_{U^*} \nabla \mathcal{D}_{U^*}^* - \epsilon) - (t_{L^*} \nabla \mathcal{D}_{L^*}^* + \epsilon) \\
&= \Delta^* - 2\epsilon.
\end{aligned}$$

Taking $\epsilon = (\Delta^*/2) - \epsilon'$ with $0 < \epsilon' < \Delta^*/2$ and $t_{j_k} > \max\{T_1, T_2\}$, we have $\Delta^{t_{j_k}} \geq \Delta^* - 2\epsilon = \Delta^* - \Delta^* + 2\epsilon' = 2\epsilon' > 0$, which contradicts $\Delta^{t_{j_k}} \to 0$. Thus, $\Delta^* \leq 0$ and the KKT conditions imply that $\overrightarrow{\alpha}^*$ is optimal. □

We remind the reader again that in the proof above, we should consider separately the updates in the positive class and those in the negative one if $z \neq 0$ in (5.30), so that we arrive to $\Delta^{t_{j_k}} = \max\{\Delta_+^{t_{j_k}}, \Delta_-^{t_{j_k}}\} > 0$ and, hence, to optimality.

6.2.4 Conclusion

We have already proved Steps 1 and 2. Thus, to finish the proof, it is only Step 3 that remains. This is easily inferred from the preliminary results of §6.2.1.

Theorem 6.8. *The iterates* $\overrightarrow{\widetilde{w}}^t = \sum \alpha_i^t t_i \widetilde{\Phi}(\overrightarrow{x}_i)$ *given by Algorithm 9 converge to the optimal* $\overrightarrow{\widetilde{w}}^*$.

Proof. Proposition 6.1 establishes that $\overrightarrow{\widetilde{w}}^*$ exists and is unique. Besides, we know by Lemma 6.7 that there is a convergent subsequence to an optimum in the dual, of the form $\left\{\overrightarrow{\alpha}^{t_j}\right\} \to \overrightarrow{\alpha}^*$.

This dual (5.43) is a continuous function of $\overrightarrow{\alpha}$, so we can also build the convergent subsequence $\left\{\mathcal{D}^{t_j}\right\} \to \mathcal{D}^*$. As we know from the proof of Proposition 6.6 that the whole sequence $\left\{\mathcal{D}^t\right\}$ is convergent, it must be the case that $\left\{\mathcal{D}^t\right\} \to \mathcal{D}^*$.

Using Lemma 6.4, the result follows, because it said that $(1/2)\left\|\overrightarrow{\widetilde{w}}^t - \overrightarrow{\widetilde{w}}^*\right\|^2 \leq \mathcal{D}^t - \mathcal{D}^*$, and the right–hand side tends to zero. □

Next, we infer some consequences that can be derived from this proof.

6.3 Further Consequences

Probably the most direct consequence that comes to mind is that, if Algorithm 9 converges, and this algorithm encompasses all the particular SMO/MDM variants by means of Table 5.7, then all these variants are shown to converge at once. In the l_2 formulations, as mentioned, only the three first bounds in (6.1) and (6.2) are applicable, so the eight possible cases in Proposition 6.5 reduce to just two, and the proof becomes somewhat simpler. For the cases where $t_L = t_U$, we must keep in mind that Δ^t is selected as the maximum between Δ_+^t and Δ_-^t by (5.60), but the scheme of the proof remains unaltered, as we already mentioned.

Regarding second order selection, it is (6.2) that holds instead of (6.1). Nevertheless, note that the first bound, corresponding to the iterations with no clipping, uses Δ^t and not Δ_2^t. Since we are using the subsequence formed by such iterations, Proposition 6.6

and Lemma 6.7 still hold, so the proof is also valid. What is more, it also holds for other selection schemes suggested in [29], such as the one which chooses a "sufficiently violating pair", that is, a pair that has a Δ^t which is at least a fixed fraction of the Δ^t obtained with the MVP.

We also mentioned in §6.1 that our proof is valid for the (RCH)–GSK and Gilbert's methods with a minor change, but so far we have not made this explicit. As a matter of fact, for these algorithms the proof is simpler, because the sequence $\{\Delta^t\}$ converges globally to zero.

Let us see this for the GSK method (recall that RCH–GSK is identical to GSK from a geometrical point of view, and that Gilbert's method is a particular case of GSK where one of the hulls is reduced to a single point). In view of (4.25) and (4.28), there are two possible values of λ: $-y_L \Delta_{y_L} / \|\overrightarrow{z}_L\|_2^2$ in non–clipped iterations, and 1 in clipped ones.

If the first value is used, (4.26) tells us that

$$\mathcal{D}^t - \mathcal{D}^{t+1} = \frac{\left(t_{L^t} \Delta^t_{t_{L^t}}\right)^2}{2\|\overrightarrow{z}^t_{L^t}\|_2^2} \geq \frac{\left(t_{L^t} \Delta^t_{t_{L^t}}\right)^2}{2D^2_{t_{L^t}}},$$

where D is defined as $D_{t_{L^t}} = \max_{p,q \in \mathcal{I}_{t_{L^t}}} \|\Phi(\overrightarrow{x}_p) - \Phi(\overrightarrow{x}_q)\|_2$, that is, the diameter of the corresponding convex hull. The inequality holds because this diameter has to be greater than $\left\|\overrightarrow{z}^t_{L^t}\right\|_2$, as $\overrightarrow{w}^t_{t_{L^t}}$ lies in this convex hull, and so does $\Phi(\overrightarrow{x}_{L^t})$.

If we have a clipped iteration, it is because $-t_{L^t} \Delta^t_{t_{L^t}} \geq \left\|\overrightarrow{z}^t_{L^t}\right\|_2^2$, so the update in the dual becomes

$$\mathcal{D}^t - \mathcal{D}^{t+1} = -\frac{\left\|\overrightarrow{z}^t_{L^t}\right\|_2^2}{2} - t_{L^t} \Delta^t_{t_{L^t}} \geq \frac{t_{L^t} \Delta^t_{t_{L^t}}}{2} - t_{L^t} \Delta^t_{t_{L^t}} = -\frac{t_{L^t} \Delta^t_{t_{L^t}}}{2},$$

so that we can write a bound analogous to (6.1) as

$$\mathcal{D}^t - \mathcal{D}^{t+1} \quad \geq \quad \min\left\{\frac{\left(\Delta^t_{t_{L^t}}\right)^2}{2D^2}, -\frac{t_{L^t} \Delta^t_{t_{L^t}}}{2}\right\}. \tag{6.3}$$

The sequence $\{D^t\}$ is thus monotonically decreasing, and it is bounded by Proposition 6.2, so it has a finite limit by Theorem 2.17. Then we have now by (6.3) that $\left\{t_{L^t} \Delta^t_{t_{L^t}}\right\}$ converges to zero.

However, the definition of Δ for GSK is different than the one for SMO/MDM, so we cannot apply Lemma 6.7 to show that the limit of $\left\{\mathcal{D}^t\right\}$ is $\mathcal{D}^*$. Fortunately, noticing that $\mathcal{D}^t - \mathcal{D}^* = (1/2)\left(\left\|\overrightarrow{w}^t\right\|_2^2 - \left\|\overrightarrow{w}^*\right\|_2^2\right)$, we can write the following [107]:

$$\begin{aligned}
\frac{1}{2}\left(\left\|\overrightarrow{w}^t\right\|_2^2 - \left\|\overrightarrow{w}^*\right\|_2^2\right) &= \frac{1}{2}\left\|\overrightarrow{w}^t - \overrightarrow{w}^*\right\|_2^2 + \left\langle\overrightarrow{w}^*, \overrightarrow{w}^t - \overrightarrow{w}^*\right\rangle \\
&\leq \left\|\overrightarrow{w}^t - \overrightarrow{w}^*\right\|_2^2 + \left\langle\overrightarrow{w}^*, \overrightarrow{w}^t - \overrightarrow{w}^*\right\rangle \\
&= \left\langle\overrightarrow{w}^t - \overrightarrow{w}^*, \overrightarrow{w}^t - \overrightarrow{w}^* + \overrightarrow{w}^*\right\rangle \\
&= \left\|\overrightarrow{w}^t\right\|_2^2 - \left\langle\overrightarrow{w}^t, \overrightarrow{w}^*\right\rangle \\
&= \left\|\overrightarrow{w}^t\right\|_2^2 - \sum_{i\in\mathcal{I}_+}\alpha_i^* t_i\left\langle\overrightarrow{w}^t, \Phi\left(\overrightarrow{x}_i\right)\right\rangle - \sum_{i\in\mathcal{I}_-}\alpha_i^* t_i\left\langle\overrightarrow{w}^t, \Phi\left(\overrightarrow{x}_i\right)\right\rangle \\
&\leq \left\|\overrightarrow{w}^t\right\|_2^2 - \min_{i\in\mathcal{I}_+}\left\{t_i\left\langle\overrightarrow{w}^t, \Phi\left(\overrightarrow{x}_i\right)\right\rangle\right\} - \min_{i\in\mathcal{I}_-}\left\{t_i\left\langle\overrightarrow{w}^t, \Phi\left(\overrightarrow{x}_i\right)\right\rangle\right\} \\
&= \left\langle\overrightarrow{w}^t, \overrightarrow{w}_+^t\right\rangle - \min_{i\in\mathcal{I}_+}\left\{t_i\left\langle\overrightarrow{w}^t, \Phi\left(\overrightarrow{x}_i\right)\right\rangle\right\} - \left\langle\overrightarrow{w}^t, \overrightarrow{w}_-^t\right\rangle - \\
&\quad \min_{i\in\mathcal{I}_-}\left\{t_i\left\langle\overrightarrow{w}^t, \Phi\left(\overrightarrow{x}_i\right)\right\rangle\right\} \\
&= -\Delta_+^t + \Delta_-^t \\
&\leq 2\Delta^t,
\end{aligned}$$

where we used the definition of $\Delta^t = \max\left\{-\Delta_+^t, \Delta_-^t\right\}$ and of $\Delta_\pm^t$, according to Algorithm 4. Note that for the RCH case, instead of decomposing $\overrightarrow{w}^*$ in its coefficients α_i, it should be decomposed in its unknown coefficients relative to the extreme points of the reduced convex hulls. It does not matter that these coefficients are unknown, because they are non–negative and we can use the minimum terms safely. For Gilbert's method, there is only one class involved, but the bound is the same.

With this bound and the fact that $\left\{\Delta^t\right\} \to 0$, we automatically have that $\left\{\overrightarrow{w}^t\right\} \to \overrightarrow{w}^*$ and $\left\{\mathcal{D}^t\right\} \to \mathcal{D}^*$, so the proof becomes considerably simpler.

Next, we move to the justification of a shrinking strategy for the general SMO algorithm. The word "shrinking" comes from the fact that this technique is aimed to remove some of the patterns from the dual problem. Removing coefficients allows in principle to solve a smaller dual, which accelerates execution time, since this time is quadratically dependent on the number of patterns. As a byproduct, it also allows to save memory space. Its main drawback is that, if the removal is not sound it will need to be undone, and the dual will have to be solved without removing these patterns.

Shrinking was first proposed by T. Joachims in [73] for the SVM^{light} algorithm, so it is also applicable to SMO. The software LIBSVM [62] is a successful example of an implementation of SMO that uses shrinking. The same team that implemented this

software showed its theoretical justification in [104]. However, as this justification is also valid for the general case of SVM^{light}, it is also somewhat cumbersome. Here we give an alternative, simpler justification building on the arguments given by Mitchell *et al.* in [28] to show that the MDM algorithm eventually operates in the optimal facets of the convex hull involved in the MNP problem.

Shrinking, which is very natural for MDM/SMO, turns out not to be valid for (RCH)–GSK. However, our justification relies on two intermediate lemmas which hold as well for GSK. It is Theorem 6.11 and Corollary 6.12 which hold exclusively for MDM/SMO. As done for the proof of convergence, we describe it for the l_1 case. It should be kept in mind that for the l_2 formulations some of the sets defined in the sequel do not apply, because the coefficients are not bounded from above by u. Besides, if $t_L = t_U$, the treatment in the sequel should be separated for the positive and negative classes.

Let us define the following sets of indices, related to the KKT conditions (5.58):

$$\begin{array}{rcl} \mathcal{H}_1 & = & \{i : t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb^*\right) = q_i - z\rho^*\}, \\ \mathcal{H}_2 & = & \{i : t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb^*\right) > q_i - z\rho^*\}, \\ \mathcal{H}_3 & = & \{i : t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb^*\right) < q_i - z\rho^*\}. \end{array} \tag{6.4}$$

What we seek to show is that, after a finite number of iterations, in Algorithm 9 we can only choose L and U in $\mathcal{H}_1$. We shall prove this after two lemmas (also valid for GSK), which we precede by the definition of the following quantities:

$$\begin{array}{rcl} \delta_2 & = & \min_{\mathcal{H}_2} \{t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb^*\right) - (q_i - z\rho^*)\}, \\ \delta_3 & = & \min_{\mathcal{H}_3} \{(q_i - z\rho^*) - t_i \left(\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle + sb^*\right)\}, \end{array}$$

together with $\delta = \min\{\delta_2, \delta_3\}$, which is obviously a positive value. With these quantities, it is straightforward to see that

$$\begin{array}{ll} \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle - t_i \left(q_i - z\rho^*\right) \geq -sb^* + \delta & \forall i \in (\mathcal{H}_2 \cap \mathcal{I}_+) \cup (\mathcal{H}_3 \cap \mathcal{I}_-), \\ \langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right)\rangle - t_i \left(q_i - z\rho^*\right) \leq -sb^* - \delta & \forall i \in (\mathcal{H}_2 \cap \mathcal{I}_-) \cup (\mathcal{H}_3 \cap \mathcal{I}_+), \end{array}$$

Besides, the inspection of the KKT conditions (5.58) makes clear that $\delta_3 = \min_{i \in \mathcal{H}_3}\{\xi_i^*\}$. Hence

$$\delta = \min\left\{\delta_2, \min_{i \in \mathcal{H}_3}\{\xi_i^*\}\right\}. \tag{6.5}$$

We can now show the first lemma:

Lemma 6.9. *The quantities* $\sum_{\mathcal{H}_2} \alpha_i^t$ *and* $\sum_{\mathcal{H}_3} \left(u - \alpha_i^t\right)$ *tend to 0 as* $t \to \infty$.

Proof. Since $\mathcal{P}^* = -\mathcal{D}^*$, we infer from (5.43) and (5.30) that $\|\overrightarrow{w}^*\|^2 = -r\rho^* - u\sum_i \xi_i^* + \sum_i \alpha_i^* q_i$.

We can define then the quantity A as

$$\begin{aligned} \left\langle \overrightarrow{w}^t - \overrightarrow{w}^*, \overrightarrow{w}^* \right\rangle &= \left\langle \overrightarrow{w}^t, \overrightarrow{w}^* \right\rangle - \|\overrightarrow{w}^*\|_2^2 \\ &= \sum_i \alpha_i^t t_i \left\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + r\rho^* + u\sum_i \xi_i^* - \sum_i \alpha_i^* q_i. \end{aligned}$$

Splitting the first summation on the different index subsets, and using the fact that $\sum_i \alpha_i t_i = 0$, the above can be written as

$$A = \sum_{\mathcal{H}_1 \cup \mathcal{H}_2 \cup \mathcal{H}_3} \alpha_i^t t_i \left(\left\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + sb^* \right) + r\rho^* + u\sum_i \xi_i^* - \sum_i \alpha_i^* q_i.$$

Applying the definition of $\mathcal{H}_1$ in (6.4) gives:

$$A = \sum_{\mathcal{H}_1} \alpha_i^t \left(q_i - z\rho^*\right) + \sum_{\mathcal{H}_2 \cup \mathcal{H}_3} \alpha_i^t t_i \left(\left\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + sb^* \right) + r\rho^* + u\sum_i \xi_i^* - \sum_i \alpha_i^* q_i.$$

If $i \in \mathcal{H}_3$, we must have $\alpha_i^* = u$. By the complementary slackness conditions of (5.58) we get $t_i \left(\left\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + sb^* \right) = q_i - z\rho^* - \xi_i^*$, so the preceding can be written as

$$\begin{aligned} A &= \sum_{\mathcal{H}_1 \cup \mathcal{H}_3} \alpha_i^t \left(q_i - z\rho^*\right) + \sum_{\mathcal{H}_2} \alpha_i^t t_i \left(\left\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + sb^* \right) - \\ &\quad \sum_{\mathcal{H}_3} \alpha_i^t \xi_i^* + r\rho^* + u\sum_i \xi_i^* - \sum_i \alpha_i^* q_i. \end{aligned}$$

For $i \in \mathcal{H}_2$, it is the case that $\alpha_i^* = 0$, and the complementary slackness conditions tell us that $\xi_i^* = 0$. We can also apply (6.5), so that $t_i \left(\left\langle \overrightarrow{w}^*, \Phi\left(\overrightarrow{x}_i\right) \right\rangle + sb^* \right) \geq q_i - z\rho^* + \delta$. This allows to write

$$A \;\geq\; \sum_i \alpha_i^t \left(q_i - z\rho^*\right) + \delta \sum_{\mathcal{H}_2} \alpha_i^t - \sum_{\mathcal{H}_3} \alpha_i^t \xi_i^* + r\rho^* + u \sum_i \xi_i^* - \sum_i \alpha_i^* q_i.$$

The terms with ρ^* cancel because $z \sum_i \alpha_i^t = r$. Rearranging the rest of terms and using the fact that ξ_i^* is non–zero only in $\mathcal{H}_3$:

$$A \;\geq\; \sum_i \left(\alpha_i^t - \alpha_i^*\right) q_i + \delta \sum_{\mathcal{H}_2} \alpha_i^t + \sum_{\mathcal{H}_3} \left(u - \alpha_i^t\right) \xi_i^*.$$

Finally, applying again (6.5) yields

$$A \;\geq\; \sum_i \left(\alpha_i^t - \alpha_i^*\right) q_i + \delta \left(\sum_{\mathcal{H}_2} \alpha_i^t + \sum_{\mathcal{H}_3} \left(u - \alpha_i^t\right) \right).$$

In the right–hand side, the first summation tends to zero because of Theorem 6.8, the global convergence of $\left\{\mathcal{D}^t\right\} \to \mathcal{D}^*$, and the dual objective function being $(1/2) \left\| \vec{\widetilde{w}} \right\|_2^2 - \sum_i \alpha_i q_i$. According to the definition of A, the left–hand side also tends to zero by Theorem 6.8, and $\delta > 0$, so the lemma holds. □

We recall once more that we are assuming throughout this chapter that t_L need not be equal to t_U because $r = z = 0$ (this is verified in Tables 5.6 and 5.7). Hence, we get in (6.4), using (5.47), that

$$\begin{aligned} t_i \nabla \mathcal{D}_i^* &\geq -sb^* + \delta \qquad \forall i \in \left(\mathcal{H}_2 \cap \mathcal{I}_+\right) \cup \left(\mathcal{H}_3 \cap \mathcal{I}_-\right), \\ t_i \nabla \mathcal{D}_i^* &\leq -sb^* - \delta \qquad \forall i \in \left(\mathcal{H}_2 \cap \mathcal{I}_-\right) \cup \left(\mathcal{H}_3 \cap \mathcal{I}_+\right). \end{aligned}$$

Otherwise we would have needed to separate the index sets in (6.4) in its subsets that belong to each of the two different classes, as was done in [108] for the RCH–MDM method. However, for the sake of simplicity we continue with this assumption, and we show the second lemma:

Lemma 6.10. *There is a T_1 such that, if $t \geq T_1$:*

$$\begin{aligned} t_i \nabla \mathcal{D}_i^t &\geq t_j \nabla \mathcal{D}_j^t + \delta/2 \qquad \forall i \in (\mathcal{H}_2 \cap \mathcal{I}_+) \cup (\mathcal{H}_3 \cap \mathcal{I}_-), j \in \mathcal{H}_1, \\ t_i \nabla \mathcal{D}_i^t &\leq t_j \nabla \mathcal{D}_j^t - \delta/2 \qquad \forall i \in (\mathcal{H}_2 \cap \mathcal{I}_-) \cup (\mathcal{H}_3 \cap \mathcal{I}_+), j \in \mathcal{H}_1, \end{aligned}$$

where δ is defined in (6.5).

Proof. We already know from Theorem 6.8 that $\overrightarrow{w}^t \to \overrightarrow{w}^*$, so that $t_i \nabla \mathcal{D}_i^t \to t_i \nabla \mathcal{D}_i^* \ \forall i$. Let T_1 be such that, for $t \geq T_1$, we have $|t_i \nabla \mathcal{D}_i^t - t_i \nabla \mathcal{D}_i^*| \leq \delta/4$.

For the first result, let $i \in (\mathcal{H}_2 \cap \mathcal{I}_+) \cup (\mathcal{H}_3 \cap \mathcal{I}_-), j \in \mathcal{H}_1$ and proceed as follows:

$$\begin{aligned} t_i \nabla \mathcal{D}_i^t &= t_i \nabla \mathcal{D}_i^t - t_i \nabla \mathcal{D}_i^* + t_i \nabla \mathcal{D}_i^* \geq -\frac{\delta}{4} + t_i \nabla \mathcal{D}_i^* \geq -\frac{\delta}{4} - sb^* + \delta \\ &= \frac{3\delta}{4} - sb^* = \frac{3\delta}{4} + t_j \nabla \mathcal{D}_j^* = \frac{3\delta}{4} + t_j \nabla \mathcal{D}_j^* - t_j \nabla \mathcal{D}_j^t + t_j \nabla \mathcal{D}_j^t \\ &\geq \frac{3\delta}{4} - \frac{\delta}{4} + t_j \nabla \mathcal{D}_j^t = \frac{\delta}{2} + t_j \nabla \mathcal{D}_j^t. \end{aligned}$$

The proof for the second result is analogous. □

Let us define the sets $\mathcal{A}^t = \{i : \alpha_i^t > 0\}$ and $\mathcal{B}^t = \{i : \alpha_i^t < u\}$. Now we are ready to show the following:

Theorem 6.11. *For Algorithm 9, there is a T_0 such that $\forall t \geq T_0$ we have $\mathcal{A}^t \subset \mathcal{H}_1 \cup \mathcal{H}_3$ and $\mathcal{B}^t \subset \mathcal{H}_1 \cup \mathcal{H}_2$.*

Proof. Let us show that $\mathcal{A}^t \subset \mathcal{H}_1 \cup \mathcal{H}_3$ for some appropriate T_0 and $t \geq T_0$. Assume to the contrary that this is not true, so that $\mathcal{A}^t \cap \mathcal{H}_2 \neq \emptyset$.

There are two possible (non–exclusive) cases: if $\mathcal{A}^t \cap \mathcal{H}_2 \cap \mathcal{I}_+ \neq \emptyset$, Lemma 6.10 and (5.50) tell us to choose U^t in $\mathcal{H}_2$. If $\mathcal{A}^t \cap \mathcal{H}_2 \cap \mathcal{I}_- \neq \emptyset$, it is L^t which is chosen in $\mathcal{H}_2$. In any case, it is clear that there is a T_1 such that $\Delta^t \geq \delta/2$ for all $t \geq T_1$, by Lemma 6.10 and (5.48).

Using (5.46), we will have from that moment T_1 on that

$$\overline{\lambda}^t = \frac{\Delta^t}{\|\overrightarrow{z}_{L^t U^t}\|_2^2} \geq \frac{\delta}{2D^2},$$

where D is defined as in (6.1). But, because of Lemma 6.9, there is another T_2 such that

$$\sum_{\mathcal{H}_2} \alpha_i^t \leq \frac{\delta}{2D^2}$$

for all $t \geq T_2$. Taking $T_3 = \max\{T_1, T_2\}$ means that $\overline{\lambda}^t$ will be clipped in (5.52), so that either $\alpha_{U^t}^{t+1}$ or $\alpha_{L^t}^{t+1}$ become equal to 0.

In other words, one coefficient α_i^t from $\mathcal{A}^t$, with $i \in \mathcal{H}_2$, leaves the set and is not present in $\mathcal{A}^{t+1}$. Therefore, in order to keep constant the size of the set $\mathcal{A}^t \cap \mathcal{H}_2$, the only possibility is that the other coefficient being updated ($\alpha_{L^t}^t$ or $\alpha_{U^t}^t$, respectively) does not belong to $\mathcal{A}^t$ (i.e. its value is zero), enters $\mathcal{A}^{t+1}$, and L^t (U^t) is in $\mathcal{H}_2$.

However, this is not possible. According to (5.50), L^t should belong to $\mathcal{I}_+$ (respectively, $U^t \in \mathcal{I}_-$). But then Lemma 6.10 tells us that this index is in $\mathcal{H}_1 \cup \mathcal{H}_3$ (the case where $\mathcal{H}_1 \cup \mathcal{H}_3 = \emptyset$ is impossible).

Thus, the size of $\mathcal{A}^t \cap \mathcal{H}_2$ decreases in every iteration, and at most after $|\mathcal{H}_2|$ iterations after T_3 this size is zero. The first part of the result follows with $T_0 = T_3 + |\mathcal{H}_2|$. The proof for $\mathcal{B}^t$ is analogous. □

The justification for shrinking in Algorithm 9 now follows:

Corollary 6.12. *There is a T_0 such that the coefficients L^t, U^t for $t \geq T_0$ can only be chosen in $\mathcal{H}_1$.*

Proof. Let us take the same T_0 as in Theorem 6.11. We know from the proof of this theorem that, for $t \geq T_0$, all coefficients $\alpha_i, i \in \mathcal{H}_2$ remain zero. The analogous proof for $\mathcal{B}^t$ tells us that all coefficients $\alpha_i, i \in \mathcal{H}_3$ eventually remain at the value u.

Since Algorithm 9 changes the value of two coefficients in every iteration, it cannot choose any point in $\mathcal{H}_2$ or $\mathcal{H}_3$ for updating. Thus, after that moment T_0, we will only be able to choose and change the coefficients in $\mathcal{H}_1$. □

As a consequence, the so–called *shrinking* strategy applied in the LIBSVM and SVM^{light} packages is also applicable and sensible in the general SMO algorithm. The strategy suggests, for any α_i^t which is equal to a bound and remains there for a while, to fix it to that bound and remove it from the problem, since it is likely that it has to be also equal to that bound in the optimum. In other words, i is likely to belong to $\mathcal{H}_2 \cup \mathcal{H}_3$, so we can assume α_i^t to be 0 or u from that moment on, in the hope that α_i^* has also that very value.

Of course, this assumption might not be correct, if either we have still not reached the iteration T_0, or if i finally belongs to $\mathcal{H}_1$, even if it has not been modified for a while and remained in bounds. In that case, the optimization should be resumed after convergence, not assuming anymore that α_i^t is fixed.

6.4 Inclusion of LS–SVMs

In this final section, we briefly discuss how the contents of the chapter can also be adapted to cover LS–SVMs, according to what was explained in the previous chapter in §5.2.3.

Regarding convergence, the proof is still valid after the changes in the general SMO method outlined in that section, but there are some places where it slightly differs from what has been described in 6.2.

Proposition 6.1 still holds for all optimal solutions $\left(\overrightarrow{w}^*, b^*, \overrightarrow{\tau}^*, \overrightarrow{\xi}^*, \rho^*\right)$. The proof is exactly the same, the only difference being that in the expression for $\psi(\delta)$ there is now an additional term

$$-l\delta \sum_i \left((\overrightarrow{\tau}_2^*)_i - (\overrightarrow{\tau}_1^*)_i \right).$$

Nevertheless, this has no influence after differentiating twice with respect to δ, so the result follows.

As for Proposition 6.2, the result also holds, because the dual objective functions of (5.66) and (5.68) are identical to the ones in (5.34) and (5.38). For the specific case of LS–SVMs, we have to use the part of the proof where $z \sum_i \alpha_i = r$ is trivial and $\overrightarrow{\alpha} = \overrightarrow{0}$ is a feasible solution.

Considering that the new clips are those in (5.72), let us examine the proof of Proposition 6.3. Here, we need to replace $\alpha_{L^t}^t$ by $\alpha_{L^t}^t - \bar{l}$. Recall from §5.2.3 that $\bar{l} = l$ when $p = 2$ formulations and $\bar{l} = \max\{l, -u\}$ when $p = 1$. Finally, it turns out that (6.1) is now further generalized to

$$\begin{aligned} \mathcal{D}^t - \mathcal{D}^{t+1} \quad \geq \quad & \min \left\{ \frac{(\Delta^t)^2}{2D^2}, \frac{\left(\alpha_{L^t}^t - \bar{l}\right)\Delta^t}{2}, \frac{\left(\alpha_{U^t}^t - \bar{l}\right)\Delta^t}{2}, \right. \\ & \left. \frac{\left(u - \alpha_{L^t}^t\right)\Delta^t}{2}, \frac{\left(u - \alpha_{U^t}^t\right)\Delta^t}{2} \right\}, \end{aligned} \tag{6.6}$$

and (6.2) to

$$\begin{aligned}\mathcal{D}^t - \mathcal{D}^{t+1} \quad \geq \quad & \min\left\{\frac{(\Delta^t)^2}{2D^2}, \frac{\left(\alpha^t_{L_2^t} - \bar{l}\right)\Delta_2^t}{2}, \frac{\left(\alpha^t_{U_2^t} - \bar{l}\right)\Delta_2^t}{2},\right. \\ & \left.\frac{\left(u - \alpha^t_{L_2^t}\right)\Delta_2^t}{2}, \frac{\left(u - \alpha^t_{U_2^t}\right)\Delta_2^t}{2}\right\}. \end{aligned} \tag{6.7}$$

Continuing with Lemma 6.4, strong duality is still valid, because (5.62) is still a convex problem with affine constraints. In this proof, we can now write for the l_1 cases:

$$\left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle = \sum_i \alpha_i^t \left(q_i + \tau_i^* - \xi_i^* - z\rho^*\right),$$

and the rest of the derivation has the form

$$\begin{aligned}\frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^t - \overrightarrow{\widetilde{w}}^*\right\|_2^2 &= \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^t\right\|_2^2 - \left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle + \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^*\right\|_2^2 \\ &= \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^t\right\|_2^2 - \sum_i \alpha_i^t \left(q_i + \tau_i^* - \xi_i^* - z\rho^*\right) + \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^*\right\|_2^2 \\ &= \mathcal{D}^t - \sum_i \alpha_i^t \left(\tau_i^* - \xi_i^* - z\rho^*\right) + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 \\ &= \mathcal{D}^t - \sum_i \alpha_i^t \left(\tau_i^* - \xi_i^*\right) + r\rho^* + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 \\ &\leq \mathcal{D}^t - \bar{l}\sum_i \tau_i^* + u\sum_i \xi_i^* + r\rho^* + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 \\ &\leq \mathcal{D}^t - l\sum_i \tau_i^* + u\left\|\overrightarrow{\xi}^*\right\|_1 + r\rho^* + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 \\ &= \mathcal{D}^t + \mathcal{P}^* \\ &= \mathcal{D}^t - \mathcal{D}^*. \end{aligned}$$

For the l_2 cases we get

$$\left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle = \sum_i \alpha_i^t \left(q_i + \tau_i^* - z\rho^*\right),$$

so that

$$
\begin{aligned}
\frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^t - \overrightarrow{\widetilde{w}}^*\right\|_2^2 &= \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^t\right\|_2^2 - \left\langle \overrightarrow{\widetilde{w}}^t, \overrightarrow{\widetilde{w}}^* \right\rangle + \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^*\right\|_2^2 \\
&= \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^t\right\|_2^2 - \sum_i \alpha_i^t \left(q_i + \tau_i^* - z\rho^*\right) + \frac{1}{2}\left\|\overrightarrow{\widetilde{w}}^*\right\|_2^2 \\
&= \mathcal{D}^t - \sum_i \alpha_i^t \left(\tau_i^* - z\rho^*\right) + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 + \frac{1}{2u}\left\|\overrightarrow{\alpha}^*\right\|_2^2 \\
&= \mathcal{D}^t - \sum_i \alpha_i^t \tau_i^* + r\rho^* + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 + \frac{1}{2u}\left\|\overrightarrow{\alpha}^*\right\|_2^2 \\
&= \mathcal{D}^t - \sum_i \alpha_i^t \tau_i^* + r\rho^* + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 + \frac{u}{2}\left\|\overrightarrow{\xi}^*\right\|_2^2 \\
&\leq \mathcal{D}^t - l\sum_i \tau_i^* + r\rho^* + \frac{1}{2}\left\|\overrightarrow{w}^*\right\|_2^2 + \frac{u}{2}\left\|\overrightarrow{\xi}^*\right\|_2^2 \\
&= \mathcal{D}^t + \mathcal{P}^* \\
&= \mathcal{D}^t - \mathcal{D}^*.
\end{aligned}
$$

For the rest of the proof of convergence, the only change that needs to be done is that in the proof of Proposition 6.5 we have to use the value $\bar{l}$ instead of 0, but the reasoning itself remains unchanged.

Similarly, the justification of shrinking remains basically unchanged. Using the new KKT conditions (5.70), it is easy to see that now instead of (6.5) we can write

$$
\delta = \min\left\{\min_{i\in\mathcal{H}_2}\left\{\tau_i^*\right\}, \min_{i\in\mathcal{H}_3}\left\{\xi_i^*\right\}\right\}, \tag{6.8}
$$

which allows to show in Lemma 6.9 that it is now the sum $\sum_{\mathcal{H}_2}\left(\alpha_i^t - \bar{l}\right)$ that tends to 0. As for Theorem 6.11, the definition of $\mathcal{A}^t$ is changed to $\mathcal{A}^t = \{i : \alpha_i^t > \bar{l}\}$.

Of course, for the particular case of LS–SVMs this shrinking strategy is not applicable, as the multipliers are unbounded and all patterns belong to $\mathcal{H}_1$.

Chapter 7

Conclusions

We have come to the end of this thesis. This chapter discusses briefly what we have learnt in the previous chapters, summarizes the contributions and gives pointers to possible extensions to the work presented here.

7.1 Discussion and Contributions

The text follows a linear structure intended to give a detailed picture of the state–of–the–art in the literature of Support Vector Machines algorithms, and paving the way for the contributions presented here. Chapters 1 and 2 can be regarded as purely introductory; the first one to the discipline of Machine Learning in general, and the second one to the mathematics that are used in the field.

Both chapters explain the main ideas upon which Support Vector Machines work. These are formulated in Chapter 3, in their different variants that have been proposed throughout the years: l_1 and l_2, Support Vector Classification and Support Vector Regression, ν–SVMs, One–Class SVMs and LS–SVMs. The chapter concludes with the MNP and NPP geometric formulations.

At that point, it seemed that all those formulations, despite having common elements, have not been approached under a unified point of view. However, Chapter 4 started to give some hints towards a unification. There, we described the SMO, GSK and MDM algorithms with a common framework that made us see points shared by these methods, such as minimizing a positive quantity Δ (in two different modalities, depending on whether we are forced to select points from the same class or not) that should be non–positive in the optimum, or trying to maximize the dual change as much as possible in every iteration. It was also pointed out how the GSK algorithm has been successfully

generalized to cover reduced convex hulls, but that the generalization that has been suggested for MDM does not work properly.

With this common framework at hand and the intuition we gained in Chapter 4, Chapter 5 is the one that provides the unification of all the formulations considered in Chapter 3. The contributions of this chapter can be summarized as follows:

- A generalization of the MDM algorithm (RCH–MDM) that works properly for reduced convex hulls.
- An illustration of the inherent relationship between the SVC, ν–SVC and NPP formulations, both for the l_1 and l_2 cases.
- A justification of why the MDM algorithm can be regarded as a particular case of the SMO method.
- An experimental comparison of the SMO, MDM and GSK algorithms that illustrates the three points already mentioned.
- A general formulation in the primal and the dual that includes as particular cases all the formulations considered up to this point.
- A general SMO approach for this general dual that includes as particular cases the SMO/MDM variants for the different formulations.

This general SMO approach is the method we work with in Chapter 6, establishing its convergence. The contributions of this chapter are:

- A complete proof of asymptotic convergence to the optimum of the general SMO approach, thus including as particular cases all the SMO/MDM variants for the different formulations.
- An explanation on why this proof is simpler than other attempts suggested in the literature.
- A minor adaptation of the proof to cover as well the GSK method.
- A theoretical justification of the shrinking strategy used in the SMO algorithm and its variants.

Next, we give some ideas of further work that can be thought of, building on the present work.

7.2 Further Work

Having a powerful and general formulation is useful not only for devising general algorithms that are valid for all the specific cases arising from the formulation, but also for realizing that these specific cases share many common ideas, even if these ideas came up from different points of view.

This fact has become obvious in this work. For instance, the main concept behind SVMs, which is finding the hyperplane with maximal margin, is very similar to the concept of finding the closest points in two convex hulls. Although these two ideas are both geometrical, margin maximization is intended for good generalization properties, whereas the Nearest Point Problem is a purely geometrical problem. In the end, the SMO algorithm, which was designed specifically for margin maximization, is analogous to the MDM method, which was designed with the concept of minimizing distances in mind.

We mentioned in §5.1.3.4 that we can take advantage of this connection, trying to visualize the inefficiencies of the MDM method, tackling with them, and then applying the result to SMO. One inefficiency is its zigzagging behavior, which has been alleviated by the cycle–breaking strategy [95, 96] and the more general momentum strategy [64, 97].

An interesting study is to adapt these strategies to the general formulation suggested in the thesis, and then checking for which formulations and datasets we get better results. It was observed in [97] that the kernel parameters play an important role in the performance of the method, but the formulation specificities may also have an important influence. A more difficult goal is to come up with a better strategy that results in further savings than the momentum one.

Another way of accelerating the training phase consists of removing unnecessary patterns from the problem. In fact, it is only the SVs that determine the optimal solution of an SVM. If we knew beforehand which are the support vectors, the rest of patterns could be safely removed from the training set, without altering the final model. This would be especially useful in large–scale learning, as many patterns could be deleted before training starts, saving a great deal of disk and memory space.

This can also be seen from a geometrical point of view. As a matter of fact, it is only the points in the nearest facets of the hulls that influence the final solution. Therefore, if we could somehow identify these facets, we would safely discard points that are not going to have any influence in the final model. Actually, the shrinking strategy can be thought of as trying to achieve this: those points that do not influence the model throughout a number of iterations are assumed to neither influence the final model.

The case of regression is more difficult to see geometrically. Although [25] gave a geometrical interpretation for it, where two convex hulls were created by adding and subtracting ϵ (the width of the ϵ–tube) to the target values, it is still an open issue to establish a relationship among ϵ–SVR, ν–SVR and NPP as clearly as in Figures 5.4 and 5.5.

Another interesting observation is that we now have a simple and convergent algorithm (the general SMO method) for solving whichever formulation we can include in the primal (5.30) or in the dual (5.43), but these formulations are not limited to the SVM world. For instance, connections between LS–SVMs and Kernel Fisher Discrimant Analysis (KFDA) [109] were pointed out in [65]. Recently, KFDA has also been seen to be equivalent [110] to a feature selection algorithm termed Quadratic Programming Feature Selection (QPFS) [111]. It may be the case that KFDA, QPFS and other problems studied in the literature of Machine Learning are also particular cases of our general formulation, something which is worth exploring.

Moving on to the convergence aspects, the suggested proof for the general SMO version is simple and easy to follow, yet it is not as general as the one in [29], in the sense that it is not valid for the SVM^{light} algorithm. The reason why it is not valid for this method is because Proposition 6.5 relies on the size of the working set being two. For larger sizes, an alternative reasoning is needed to show that clipped iterations cannot go on indefinitely, but hopefully not a complex counting argument like the one by Lin. We are currently working on this topic, as well as on trying to generalize our arguments to cover the momentum–based version of SMO in [64].

In addition, it is still not clear what is the rate of convergence of the standard SMO algorithm. Under some restrictive assumptions, not usually fulfilled in practice, this rate is shown to be linear (cf. Definition 2.20) for l_1–SVC in [29]. These assumptions are two: the positive definiteness of the Gram matrix Ω, and what is termed as "non–degeneracy", which assumes that all support vectors lying on the support hyperplanes must have $0 < \alpha_i^* < C$. By inspection of (5.20), this is not normally so; a point lying on its corresponding support hyperplane can have such a coefficient, but it can also be the case that $\alpha_i^* = 0$ or $\alpha_i^* = C$.

The non–degeneracy assumption is designed so that all coefficients that should be either 0 or C in the solution are fixed from a given moment on. Observe that this is stronger than Corollary 6.12, which does not ensure that the points in $\mathcal{H}_1$ do not eventually end up in bounds. If we assume that is the case, from a given moment on we are always selecting points from $\mathcal{H}_1$ and their associated coefficients cannot reach the bounds, so no iteration is going to be clipped and the convergence rate is not difficult to establish.

A very recent work [112] removes these assumptions and concludes that when the Gram matrix is positive definite, the convergence rate is linear and that, when it is positive semidefinite, it is sublinear. However, the proof is quite complex and relies on a construct called "delay sequences", whose significance in terms of the algorithm is not clear. We will try to use some of these ideas for establishing in a simpler way the rate of convergence of the general version of SMO.

Chapter 8

Conclusiones

Con esto hemos llegado al final de la tesis. Este capítulo discute brevemente lo que hemos aprendido en los capítulos anteriores, además de resumir las contribuciones y proporcionar ideas para posibles extensiones del trabajo aquí presentado.

8.1 Discusión y Contribuciones

El texto sigue una estructura lineal con el fin de dar una visión detallada del estado del arte de la literatura sobre algoritmos de entrenamiento de Máquinas de Vectores de Soporte (SVMs), allanando de esta forma el camino para llegar a las contribuciones que enumeramos en este capítulo. Los Capítulos 1 y 2 se pueden ver como meramente introductorios; el primero a la disciplina del Aprendizaje Automático en general, y el segundo a las matemáticas que se usan en este campo.

Ambos capítulos explican las ideas principales en que se basan las SVMs. Dichas máquinas se formulan en el Capítulo 3, en las distintas variantes que se han ido proponiendo a lo largo de los años: l_1 y l_2, Clasificación mediante Vectores de Soporte (SVC), Regresión mediante Vectores de Soporte (SVR), Máquinas de Vectores de Soporte ν (ν–SVMs), Máquinas de Vectores de Soporte de Una Clase (One–Class SVMs) y Máquinas de Vectores de Soporte de Mínimos Cuadrados (LS–SVMs). El capítulo termina con las formulaciones geométricas del Problema de Norma Mínima (MNP) y del Problema de Puntos Más Cercanos (NPP).

Llegados a ese punto, parecía que todas estas formulaciones, a pesar de tener elementos en común, no se han estudiado bajo un punto de vista unificado. Sin embargo, el Capítulo 4 empieza a dar algunas ideas con vistas a una unificación. Ahí describimos los algoritmos SMO, GSK y MDM mediante un marco común que nos hace ver puntos

compartidos por estos métodos, como por ejemplo el minimizar una cantidad positiva Δ (de dos maneras distintas, según que estemos obligados a seleccionar puntos de la misma clase o no) que debería ser no positiva en el óptimo, o como intentar maximizar el cambio en la función dual lo máximo posible en cada iteración. También se hizo hincapié en cómo el algoritmo GSK se generalizó con éxito para incluir el caso de envolventes convexas reducidas, mientras que la generalización que se sugirió para MDM con el mismo fin no funciona correctamente.

Con este marco común y la intuición dada por el Capítulo 4, el Capítulo 5 es el que proporciona la unificación de todas las formulaciones consideradas en el Capítulo 3. Las contribuciones de este capítulo se pueden resumir en los siguientes puntos:

- Una generalización del algoritmo MDM (RCH–MDM) que funciona de manera adecuada para envolventes convexas reducidas.
- Una ilustración de la relación inherente entre los problemas de SVC, ν–SVC y NPP, tanto en el caso l_1 como en el l_2.
- Una justificación de por qué el algoritmo MDM puede considerarse como un caso particular del método SMO.
- Una comparación experimental de los algoritmos SMO, MDM y GSK, que ilustra los tres puntos ya mencionados.
- Una formulación general en el primal y en el dual, que incluye como casos particulares todas las formulaciones estudiadas hasta ese momento.
- Un enfoque general de tipo SMO para este dual general, que incluye como casos particulares las variantes SMO/MDM para las distintas formulaciones.

Este enfoque general de tipo SMO es el algoritmo con el que trabajamos en el Capítulo 6, donde se establece su convergencia. Las contribuciones de este capítulo son:

- Una demostración completa de la convergencia asintótica al óptimo del enfoque general SMO, incluyendo por tanto como casos particulares todas las variantes SMO/MDM específicas de las distintas formulaciones.
- Una explicación de por qué esta demostración es más sencilla que otros intentos que se han llevado a cabo en la literatura.
- Una ligera adaptación de la demostración para cubrir también el algoritmo GSK.

- Una justificación teórica de la estrategia de *shrinking* utilizada en el algoritmo SMO y sus variantes.

A continuación damos algunas ideas de trabajo futuro que se basan en el trabajo presente.

8.2 Trabajo Futuro

El tener una formulación potente y general no sólo es útil para diseñar algoritmos genéricos que sean válidos para todos los casos específicos de dicha formulación, sino también para darse cuenta de que estos casos específicos comparten muchas ideas comunes, aun cuando estas ideas surgieron a partir de diferentes puntos de vista.

La tesis deja claro este hecho. Por ejemplo, el concepto base de las SVMs, que es el de encontrar el hiperplano de margen máximo, es muy similar al concepto de encontrar los dos puntos más cercanos de dos envolventes convexas. Aunque estas dos ideas son ambas geométricas, la maximización del margen tiene como propósito el generalizar bien, mientras que el Problema de Puntos Más Cercanos es un problema puramente geométrico. Al final, el algoritmo SMO, que se diseñó específicamente con vistas a la maximización del margen, es análogo al algoritmo MDM, el cual se diseñó con la idea de minimizar distancias.

Hemos mencionado en §5.1.3.4 que es posible aprovechar esta conexión, intentando visualizar las ineficiencias del algoritmo MDM, enfrentándonos a ellas, y aplicando el resultado a SMO. Una ineficiencia es su comportamiento zigzagueante, el cual se suaviza mediante la estrategia de "rotura de ciclos" [95, 96] y mediante una estrategia más general basada en el momento [64, 97].

Un estudio interesante consistiría en adaptar estas estrategias a la formulación general propuesta en esta tesis, y comprobar para qué formulaciones y conjuntos de datos se consiguen mejores resultados. En [97] se observó que los parámetros del núcleo desempeñan un papel importante en el rendimiento del método, pero las especificidades de la formulación quizás tengan también una influencia considerable. Una meta más ambiciosa es la de llegar a una estrategia mejor, que tenga como resultado mejoras más cuantiosas que las obtenidas con el momento.

Otra manera de acelerar la fase de entrenamiento consistiría en eliminar del problema aquellos patrones que sean innecesarios. De hecho, los vectores de soporte son los únicos que determinan la solución óptima de una SVM. Si supiéramos de antemano cuáles son esos vectores de soporte, los demás patrones podrían eliminarse de forma segura del

conjunto de entrenamiento, sin alterar el modelo final. Esto sería especialmente útil en aprendizaje a gran escala, puesto que muchos patrones podrían eliminarse de partida antes de empezar el entrenamiento, ahorrando así una gran cantidad de espacio en disco y en memoria.

Esto también puede verse desde un punto de vista geométrico. De hecho, los únicos puntos que influyen en la solución final son aquéllos situados en las caras más próximas de las envolventes. Así, si pudiéramos identificar de alguna forma cuáles son esas caras, podríamos descartar de forma segura los puntos que no van a tener ninguna influencia en el modelo final. Es más, puede considerarse que la estrategia de *shrinking* intenta perseguir este fin: supone que aquellos puntos que no influyan en el modelo durante un número de iteraciones tampoco lo van a hacer en el modelo final.

El caso de regresión es más difícil de ver geométricamente. Aunque [25] dio una interpretación geométrica para esta formulación, en la que se crean dos envolventes convexas mediante la suma y la resta de ϵ (la anchura del tubo–ϵ) a los valores objetivo, todavía es un problema abierto el establecer una relación entre la Regresión Mediante Vectores de Soporte ϵ, la Regresión Mediante Vectores de Soporte ν y el Problema de Puntos Más Cercanos que sea tan clara como la de las Figuras 5.4 y 5.5.

Otra observación interesante es que ahora tenemos un algoritmo sencillo y convergente (el método SMO general) para resolver cualquier formulación que podamos incluir en el primal (5.30) o en el dual (5.43), pero estas formulaciones no se limitan al mundo de las SVMs. Por ejemplo, se han expuesto conexiones entre las LS–SVMs y el Análisis Discriminante de Fisher con Núcleos (KFDA) [109] en [65]. Recientemente, el KFDA también se ha visto que es equivalente [110] a un algoritmo de selección de características llamado Selección de Características mediante Programación Cuadrática (QPFS) [111]. Podría ser el caso de que KFDA, QPFS y otros problemas estudiados en la literatura de Aprendizaje Automático fueran también casos particulares de nuestra formulación general, algo que merece la pena estudiar.

Pasando a aspectos de convergencia, la demostración propuesta para la versión general de SMO es sencilla y fácil de seguir, pero no es tan general como la demostración de [29], en el sentido de que no es válida para el algoritmo SVM^{light}. La razón por la que no es válida para este método es porque la Proposición 6.5 se basa en que el tamaño del *working set* es de dos. Para tamaños mayores, se necesita un razonamiento alternativo para demostrar que las iteraciones con recorte no se pueden prolongar indefinidamente, pero con suerte no un argumento complejo de conteo como el de Lin. En este momento estamos trabajando en este tema, así como en intentar generalizar nuestros argumentos para cubrir la versión de SMO basada en el momento [64].

Además, no está claro todavía cuál es la tasa de convergencia del algoritmo SMO estándar. Haciendo algunas suposiciones restrictivas, que normalmente no se cumplen en la práctica, se ha visto en [29] que esta tasa es lineal (ver la Definición 2.20) para l_1–SVC. Estas suposiciones son dos: el que la matriz de Gram Ω sea definida positiva, y lo que se conoce como "non–degeneracy", que asume que todos los vectores de soporte que se encuentran sobre los hiperplanos de soporte tienen que tener $0 < \alpha_i^* < C$. Mirando (5.20), esto normalmente no es así; un punto que esté sobre su correspondiente hiperplano de soporte puede tener un coeficiente así, pero también puede darse que $\alpha_i^* = 0$ ó $\alpha_i^* = C$.

Esta suposición de *non–degeneracy* está concebida para que todos los coeficientes que tengan que ser 0 ó C en la solución estén fijos a esos valores a partir de un cierto momento. Obsérvese que esto es más fuerte que el Corolario 6.12, que no asegura que los puntos en $\mathcal{H}_1$ no terminen eventualmente en cota. Si asumimos que tal es el caso, a partir de un cierto momento siempre estaremos seleccionando puntos de $\mathcal{H}_1$ y sus coeficientes asociados no pueden alcanzar las cotas, con lo cual ninguna iteración será recortada y la tasa de convergencia no es difícil de establecer.

Un trabajo muy reciente [112] elimina estas suposiciones y concluye que cuando la matriz de Gram es definida positiva la tasa de convergencia es lineal, y que cuando es semidefinida positiva es sublineal. Sin embargo, la demostración es bastante compleja y se basa en unas entidades conocidas como "sucesiones delay", cuyo significado en términos del algoritmo no está claro. Intentaremos utilizar algunas de estas ideas para determinar de una manera más simple la tasa de convergencia de la versión general de SMO.

Appendix A

List of Publications

Here we give a summary of the publications by the author. Some are directly related to the topics covered, while some others are byproducts or centered in other lines of research. We group the publications by type.

The journal papers are the following:

- [92] explains the RCH–GSK algorithm and Tao's version of RCH–MDM for solving the RCH–NPP formulation. It also points out that the latter does not always arrive to the optimal solution in a similar way than we did here. Later on, it is suggested to use the clipping strategy of the SMO algorithm for generalizing GSK and MDM to RCH–NPP. While this works for MDM, giving rise to our version of RCH–MDM, it is shown that this is not the case for GSK.

- [67] explores the SMO version for LS–SVMs, comparing the first order and second order selections for the Gaussian kernel. It turns out that second order is nearly always more convenient than first order. Convergence is also established in a simpler way than in Chapter 6, because there are no clipped iterations, as the coefficients are unbounded. This also allows to establish the linear rate of convergence as in [29].

- [96] describes the cycle–breaking strategy for the MDM and SMO algorithms. Experimental results show that some noticeable savings can be achieved in several well–known datasets, when compared to their original versions.

- [113] suggests a refined grid search that zooms in in an intelligent way such that points are reused. It is applied for searching the SVM C and ϵ parameters, as well as the width σ of the Gaussian kernel.

Regarding conference papers we have:

- [93] proves in a slightly different way than we did here that the MDM algorithm is a particular version of SMO. Experiments to show this are performed for the l_1 and l_2 cases, analogously to the ones described in Chapter 5 for the l_1 case.
- [94] suggests a variation of the MDM algorithm where the four indexes L_+, L_-, U_+ and U_- are used in a joint update in the two classes at the same time, instead of choosing the one with largest Δ. The resulting optimization is also analytically solvable and faster than standard MDM in some of the datasets considered.
- [114] presents a preliminary proof of convergence for the MDM algorithm, and for SMO in the l_2 and hard–margin cases, which is similar in spirit to the one of Chapter 6. However, this proof did not cover the l_1 case.
- [107] changes the proof of convergence of [114] to build upon Gilbert's proof and generalizes its arguments to cover the GSK, MDM and SMO methods. Still it is only the l_2 case that was considered.
- [97] applies the momentum strategy of [64] to LS–SVMs. Remarkable savings are obtained in most of the datasets tested.
- [115] described an adaptation of the MDM algorithm for the so–called "scaled convex hulls". These are an alternative to reduced convex hulls suggested by [116], and have the advantage that their shape remains constant as they are reduced.
- [108] presents the proof of convergence of Chapter 6 for the particular cases of the RCH–GSK and RCH–MDM methods, as well as justifyling the shrinking strategy for the latter.
- [95] describes the zigzagging behavior of MDM and proposes the cycle–breaking strategy to solve it. This paper was extended in [96] to cover as well the SMO method.
- [98] explains how Rosen's gradient projection method, which multiplies the gradient by a matrix that ensures that the resulting direction is feasible and improving, can be used in SVM training without an overly expensive computational cost, due to the simple structure of the problem.
- [117] suggests a slight variation of LS–SVMs that uses the l_1–loss. As a result, the problem obtained is the same as in standard LS–SVMs, except for the fact that the coefficients lie in the interval $[-C, C]$. Thus, this formulation is also included in the general formulation of Chapter 5.

- [118] studies the possibility of minimizing the l_0–norm of LS–SVMs to obtain a much sparser solution that has comparable generalization properties.
- [119] introduced the refined grid search of [113] for finding the optimal parameters of a multilayer perceptron.
- [91] suggested the idea of clipping for solving the RCH–NPP formulation. This paper was naturally continued and extended in [92].
- [120] explores the l_2 variant of ϵ–SVR, and its application in a wind energy forecasting problem.

Bibliography

[1] G. Rätsch. *Benchmark Repository*, 2000. Datasets available at http://ida.first.fhg.de/projects/bench/benchmarks.htm.

[2] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Adaptive Comunication and Machine Learning. MIT Press, Cambridge, Massachussets, 2002.

[3] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, 2003.

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Computer Science and Scientific Computing. Academic Press, Boston, 2nd edition, 1990.

[5] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience. Wiley & Sons, New York, 2nd edition, 2001.

[6] V. Franc. *Optimization Algorithms for Kernel Methods*. PhD thesis, Center for Machine Perception, Czech Technical University, 2005.

[7] T. Glasmachers. *Gradient Based Optimization of Support Vector Machines*. PhD thesis, Institut für Neuroinformatik, Ruhr–Universität Bochum, 2008.

[8] D.B. Rubin and R.J.A. Little. *Statistical Analysis with Missing Data*. Wiley, 2002.

[9] K.P. Bennett and O.L. Mangasarian. Robust Linear Programming Discrimination of Two Linearly Inseparable Sets. *Optimization Methods and Software*, 1:23–34, 1992.

[10] V.N. Vapnik. *Statistical Learning Theory*. Adaptive and Learning Systems for Signal Processing, Communications and Control. John Wiley and Sons, New York, 1998.

[11] H. Chernoff. A Measure of Asymptotic Efficiency of Tests of a Hypothesis based on the Sum of Observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[12] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[13] V.N. Vapnik and A. Chervonenkis. The Necessary and Sufficient Conditions for Consistency in the Empirical Risk Minimization Method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

[14] A.N. Tikhonov and V.Y. Arsenin. *Solution of Ill-Posed Problems.* Winston, Washington D.C., 1977.

[15] F. Riesz and B.S. Nagy. *Functional Analysis.* Frederick Unger Publishing Co., 1995.

[16] V.A. Morozov. *Methods for Solving Incorrectly Posed Problems.* Springer Verlag, 1984.

[17] P.L. Bartlett and S. Ben-David. Hardness Results for Neural Network Approximation Problems. In *Proceedings of the 4th European Conference on Computational Learning Theory*, pages 50–62, 1999.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, New York, 2nd edition, 2009.

[19] C.M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006.

[20] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6):386–408, 1958.

[21] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[22] J.A.K. Suykens and J. Vandewalle. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[23] B. Schölkopf, A.J. Smola, R.C. Williamson, and P.L. Bartlett. New Support Vector Algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

[24] 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.

[25] J. Bi and K.P. Bennett. A Geometric Approach to Support Vector Regression. *Neurocomputing*, 55:79–108, 2003.

[26] J.C. Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In *Advances in Kernel Methods: Support Vector Learning*, pages 185–208, Cambridge, MA, USA, 1999. MIT Press.

[27] V. Franc and V. Hlavăc. An Iterative Algorithm Learning the Maximal Margin Classifier. *Pattern Recognition*, 36:1985–1996, 2003.

[28] B.F. Mitchell, V.F. Dem'yanov, and V.N. Malozemov. Finding the Point of a Polyhedron Closest to the Origin. *SIAM Journal on Control*, 12:19–26, 1974.

[29] P.-H. Chen, R.-E. Fan, and C.-J. Lin. A Study on SMO-type Decomposition Methods for Support Vector Machines. *IEEE Transactions on Neural Networks*, 17:893–908, 2006.

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

[31] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[32] J. Mercer. Functions of Positive and Negative Type and their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society, London*, A(209):415–446, 1909.

[33] D. Hilbert. Grundzüge einer Allgemeinen Theorie der Linearen Integralgleichungen. In *Nachrichten der Göttinger Akademie der Wissenschaften, Mathematisch-Physikalische Klasse*, pages 49–91, 1904.

[34] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, (68):337–404, 1950.

[35] C.A. Micchelli. Algebraic Aspects of Interpolation. *Proceedings of Symposia in Applied Mathematics*, 36:81–102, 1986.

[36] M. Krein and D. Milman. On Extreme Points of Regular Convex Sets. *Studia Mathematica*, 9:133–138, 1940.

[37] 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.

[38] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, New York, 1989.

[39] R.T. Rockafellar. *Convex Analysis*, volume 28 of *Princeton Mathematics Series*. Princeton University Press, 1970.

[40] D.G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison–Wesley, Reading, MA, 1973.

[41] W. Karush. Minima of Functions of Several Variables with Inequalities as Side Constraints. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[42] H.W. Kuhn and A.W. Tucker. Nonlinear Programming. In *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics*, pages 481–492. University of California Press, 1951.

[43] T.T. Friess. Support Vector Networks: the Kernel Adatron with Bias and Soft-Margin. *Univ. Sheffield Tech. Rep.*, 1998.

[44] S.S. Keerthi. Efficient Tuning of SVM Hyperparameters using Radius/margin Bound and Iterative Algorithms. *IEEE Transactions on Neural Networks*, 13(5):1225–1229, 2002.

[45] F. Friedrichs and C. Igel. Evolutionary Tuning of Multiple SVM Parameters. *Neurocomputing*, 64:107–117, 2005.

[46] O. Chapelle, V.N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing Multiple Parameters for Support Vector Machines. *Machine Learning*, 46(1-3):131–159, 2002.

[47] D.J. Crisp and C.J.C. Burges. A Geometric Interpretation of ν–SVM Classifiers. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

[48] C.-C. Chang and C.-J. Lin. Training ν-Support Vector Regression: Theory and Algorithms. *Neural Computation*, 14(8):1959–1977, 2002.

[49] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[50] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A Fast Procedure for computing the Distance between Complex Objects in Three Dimensional Space. *IEEE Journal on Robotics and Automation*, 4:193–203, 1988.

[51] E. Polak. *Computational Methods in Optimization*. Academic Press, 1971.

[52] W.C. Davidon. Variable Metric Method for Minimization. *SIAM Journal on Optimization*, 1:1–17, 1991.

[53] C.G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970.

[54] M. Grant and S. Boyd. CVX: Matlab Software for Disciplined Convex Programming.

[55] The MOSEK Otimization Software. Software available at http://www.mosek.com/.

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

[57] E. Osuna, R. Freund, and F. Girosi. An Improved Training Algorithm for Support Vector Machines. *Proc. 1997 IEEE Workshop*, pages 276–285, 1997.

[58] J. López. On the Relationship among the MDM, SMO and SVM-Light Algorithms for Training Support Vector Machines. Master's thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid, 2008.

[59] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, 13(3):637–649, 2001.

[60] 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.

[61] T. Glasmachers and C. Igel. Second Order SMO Improves SVM Online and Active Learning. *Neural Computation*, 20(2):374–382, 2008.

[62] C.-C. Chang and C.-J. Lin. *LIBSVM: a Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[63] T. Glasmachers and C. Igel. Maximum-Gain Working Set Selection for SVMs. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

[64] Á. Barbero. *Efficient Optimization Methods for Regularized Learning: Support Vector Machines and Total-Variation Regularization*. PhD thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid, 2011.

[65] S.S. Keerthi and S.K. Shevade. SMO Algorithm for Least-Squares SVM Formulations. *Neural Computation*, 15(2):487–507, 2003.

[66] L. Bo, L. Jiao, and L. Wang. Working Set Selection Using Functional Gain for LS-SVM. *IEEE Transactions on Neural Networks*, 18(5):1541–1544, 2007.

[67] J. López and J.A.K. Suykens. First and Second Order SMO Algorithms for LS-SVM classifiers. *Neural Processing Letters*, 33(1):31–44, 2011.

[68] A.J. Smola and B. Schölkopf. A Tutorial on Support Vector Regression. *Statistics and Computing*, 48:199–222, 2003.

[69] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, and K.R.K. Murthy. Improvements to SMO Algorithm for SVM Regression. *IEEE Transactions on Neural Networks*, 11(5):1188–1193, 2000.

[70] G.W. Flake and S. Lawrence. Efficient SVM Regression Training with SMO. *Machine Learning*, 46:271–290, 2002.

[71] C.-J. Lin. On the Convergence of the Decomposition Method for Support Vector Machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, November 2001.

[72] C.-C. Chang and C.-J. Lin. Training ν-Support Vector Classifiers: Theory and Algorithms. *Neural Computation*, 13(9):2119–2147, 2001.

[73] T. Joachims. Making Large-Scale Support Vector Machine Learning Practical. In *Advances in Kernel Methods: Support Vector Learning*, pages 169–184, Cambridge, MA, USA, 1999. MIT Press.

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

[75] MINOS Solver for Nonlinear Programming. Software available at http://www.aimms.com/features/solvers/minos.

[76] R.J. Vanderbei. An Interior Point Code for Quadratic Programming. *Optimization Methods and Software*, 12:451–484, 1999.

[77] L. Palagi and M. Sciandrone. On the Convergence of a Modified Version of svm^{light} Algorithm. *Optimization Methods and Software*, 20(2 & 3):317–334, 2005.

[78] S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone. A Convergent Decomposition Algorithm for Support Vector Machines. *Computational Optimization and Applications*, 38(2):217–234, 2007.

[79] R. Collobert and S. Bengio. SVMTorch: Support Vector Machines for Large-Scale Regression Problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[80] S.J. Wright. *Primal-Dual Interior Point Methods*. SIAM, 1997.

[81] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated Sub-Gradient Solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*, pages 807–814, 2007.

[82] V. Franc and S. Sonnenburg. Optimized Cutting Plane Algorithm for Support Vector Machines. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*, pages 320–327, 2008.

[83] T. Joachims. Training Linear SVMs in Linear Time. In *Proceedings of the 12th ACM SIGKDD Interational Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 217–226, 2006.

[84] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[85] E.G. Gilbert. Minimizing the Quadratic Form on a Convex Set. *SIAM Journal on Control*, 4:61–79, 1966.

[86] 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.

[87] M.E. Mavroforakis, M. Sdralis, and S. Theodoridis. A Geometric Nearest Point Algorithm for the Efficient Solution of the SVM Classification Task. *IEEE Transactions on Neural Networks*, 18(5):1545–1549, September 2007.

[88] 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.

[89] Q. Tao, G. W. Wu, and J. Wang. A Generalized S-K Algorithm for Learning ν-SVM Classifiers. *Pattern Recognition Letters*, 25(10):1165–1171, 2004.

[90] K.P. Bennett and E.J. Bredensteiner. Geometry in Learning. In *Geometry at Work*, 1997.

[91] J. López, Á. Barbero, and J.R. Dorronsoro. Simple Clipping Algorithms for Reduced Convex Hull SVM Training. In *Lecture Notes in Computer Science: Hybrid Artificial Intelligent Systems*, volume 5271, pages 369–377, 2008.

[92] J. López, Á. Barbero, and J.R. Dorronsoro. Clipping Algorithms for Solving the Nearest Point Problem over Reduced Convex Hulls. *Pattern Recognition*, 44(3):607–614, 2011.

[93] J. López, Á. Barbero, and J.R. Dorronsoro. On the Equivalence of the SMO and MDM Algorithms for SVM Training. In *Lecture Notes in Computer Science: Machine Learning and Knowledge Discovery in Databases*, volume 5211, pages 288–300. Springer, 2008.

[94] Á. Barbero, J. López, and J.R. Dorronsoro. A 4–Vector MDM Algorithm for Support Vector Training. In *Lecture Notes in Computer Science: Artificial Neural Networks*, volume 5165, pages 315–324, Berlin, Heidelberg, 2008. Springer-Verlag.

[95] Á. Barbero, J. López, and J.R. Dorronsoro. An Accelerated MDM Algorithm for SVM Training. In *Proceedings of the 16th European Symposium on Artificial Neural Networks*, pages 421–426, 2008.

[96] Á. Barbero, J. López, and J.R. Dorronsoro. Cycle-breaking Acceleration of SVM Training. *Neurocomputing*, 72(7-9):1398–1406, 2009.

[97] J. López, Á. Barbero, and J.R. Dorronsoro. Momentum Acceleration of Least–Squares Support Vector Machines. In *Lecture Notes in Computer Science: Artificial Neural Networks and Machine Learning - ICANN 2011*, volume 6792, pages 135–142, 2011.

[98] J. López and J.R. Dorronsoro. Rosen's Projection Method for SVM training. In *Proceedings of the 17th European Symposium on Artificial Neural Networks*, pages 183–188, 2009.

[99] S.S. Keerthi and E.G. Gilbert. Convergence of a Generalized SMO Algorithm for SVM Classifier Design. *Machine Learning*, 46:351–360, 2002.

[100] N. Takahashi and T. Nishi. Rigorous Proof of Termination of SMO Algorithm for Support Vector Machines. *IEEE Transactions on Neural Networks*, 16(3):774–776, 2005.

[101] C.-W. Hsu and C.-J. Lin. A Simple Decomposition Method for Support Vector Machines. *Machine Learning*, 46:291–314, 2002.

[102] C.-C. Chang, C.-W. Hsu, and C.-J. Lin. The Analysis of Decomposition Methods for Support Vector Machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008, 2000.

[103] C.-J. Lin. Asymptotic Convergence of an SMO Algorithm without any Assumptions. *IEEE Transactions on Neural Networks*, 13(1):248–250, January 2002.

[104] C.-J. Lin. A Formal Analysis of Stopping Criteria of Decomposition Methods for Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(5):1045–1052, 2002.

[105] N. Takahashi and T. Nishi. Global Convergence of SMO Algorithm for Support Vector Regression. *IEEE Transactions on Neural Networks*, 19(6):971–982, 2008.

[106] C.J.C. Burges and D.J. Crisp. Uniqueness Theorems for Kernel Methods. *Neurocomputing*, 55(1-2):187–220, 2003.

[107] J. López and J.R. Dorronsoro. A Common Framework for the Convergence of the GSK, MDM and SMO algorithms. In *Lecture Notes in Computer Science: Artificial Neural Networks - ICANN 2010*, volume 6353, pages 82–87, 2010.

[108] J. López and J.R. Dorronsoro. Convergence of Algorithms for Solving the Nearest Point Problem in Scaled Convex Hulls. In *International Joint Conference on Neural Networks (IJCNN'11)*, pages 413–420, 2011.

[109] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.R. Mullers. Fisher Discriminant Analysis with Kernels. In *Proceedings of the IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[110] I. Rodriguez-Luján, R. Huerta, C. Elkan, and C. Santa-Cruz. On the Equivalence of Kernel Fisher Discriminant Analysis and Kernel Quadratic Programming Feature Selection. *Pattern Recognition Letters*, 32:1567–1571, 2011.

[111] I. Rodriguez-Luján, R. Huerta, C. Elkan, and C. Santa-Cruz. Quadratic Programming Feature Selection. *Journal of Machine Learning Research*, 11:1491–1516, 2010.

[112] N. List and H.U. Simon. Svm-Optimization and Steepest-Descent Line Search. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2009)*, 2009.

[113] Á. Barbero, J. López, and J.R. Dorronsoro. Finding Optimal Model Parameters by Deterministic and Annealed Focused Grid Search. *Neurocomputing*, 72(13-15):2824–2832, 2009.

[114] J. López and J.R. Dorronsoro. A Simple Proof of the Convergence of the SMO Algorithm for Linearly Separable Problems. In *Lecture Notes in Computer Science: Artificial Neural Networks - ICANN 2009*, volume 5768, pages 904–912, 2009.

[115] J. López, Á. Barbero, and J.R. Dorronsoro. An MDM Solver for the Nearest Point Problem in Scaled Convex Hulls. In *International Joint Conference on Neural Networks (IJCNN'10)*, pages 1–8, 2010.

[116] Z. Liu, J.G. Liu, C. Pan, and G. Wang. A Novel Geometric Approach to Binary Classification Based on Scaled Convex Hulls. *IEEE Transactions on Neural Networks*, 20(7):1215–1220, 2009.

[117] J. López and J.R. Dorronsoro. Least 1-Norm SVMs: a New SVM Variant between Standard and LS-SVMs. In *Proceedings of the 18th European Symposium on Artificial Neural Networks*, pages 135–140, 2010.

[118] J. López, K. De Brabanter, J.A.K. Suykens, and J.R. Dorronsoro. Sparse LS–SVMs with l_0 Norm Minimization. In *Proceedings of the 19th European Symposium on Artificial Neural Networks*, pages 189–194, 2011.

[119] Á. Barbero, J. López, and J.R. Dorronsoro. Finding Optimal Model Parameters by Discrete Grid Search. In *Advances in Soft Computing: Innovations in Hybrid Intelligent Systems*, volume 44, pages 120–127. Springer, 2008.

[120] Á. Barbero, J. López, and J.R. Dorronsoro. Square Penalty Support Vector Regression. In *Lecture Notes in Computer Science: Intelligent Data Engineering and Automated Learning - IDEAL 2007*, pages 537–546. Springer, 2007.