Trabajo Fin de Máster

Constrained Support Vector Machines Theory and Applications to Health Science

Presented by:

Sandra Benítez Peña

Supervisors:

Dr. Rafael Blanquero Bravo, Universidad de Sevilla

Dr. Emilio Carrizosa Priego, Universidad de Sevilla

External Supervisor:

Dr. Pepa Ramírez Cobo, Universidad de Cádiz

UNIVERSIDAD D SEVILLA

June 24, 2016

Agradecimientos

En primer lugar, me gustaría dar las gracias a los doctores Emilio Carrizosa Priego y Rafael Blanquero Bravo. Gracias a ellos, estoy aquí ahora. Y gracias a ellos, estoy haciendo lo que me gusta. Creo que no existen palabras suficientes para agradecerles todo lo que han hecho por mí, tanto por lograr que se vaya haciendo realidad el sueño que he tenido desde pequeña como por lo infinitamente pacientes, comprensivos y buenos que han sido conmigo, así como por lo mucho que he podido aprender de ellos en estos últimos años.

También, dar las gracias a la doctora Pepa Ramírez Cobo, a quien he conocido este año y me ha ayudado a más no poder, demostrando la grandísima profesional y persona que es.

Como no, agradecerle a mis padres, Emilia y Hermenegildo, todos los esfuerzos que han hecho por lograr que poco a poco vaya alcanzando mi meta. A mi hermana, Raquel, por ser mi compañera de rabietas mientras estudiabamos, y siempre confiar en mi. A Juan Manuel, por ser el que (quizás) más ha tenido que aguantarme. Por siempre ayudarme a seguir para adelante, confiar tanto en mi, y estar siempre a mi vera.

Contents

Chapter 1

Introduction

Recently, data mining has become a very important tool for handling data and also for discovering patterns in order to produce information for decision-making. One of the most important tasks in data mining is supervised classification, which has been successfully applied in many different fields such as biology or medicine.

The focus of this work is on Support Vector Machines (SVMs) classifiers, which were first introduced by Vapnik in early 90s and right now they are one of the main exponents in supervised classification.

This work is structured in three chapters apart from this introduction. In Chapter 2, we provide the necessary theory about general classification. In Chapter 3, SVMs theory is introduced, focusing on the two-class classification problem. Then, a brief explanation of the theory behind the classification of more than two classes (also called multiclass classification) is presented. After that, also in Chapter 3, some applications of SVMs are shown, using the R package e1071 and AMPL. Then, in Chapter 4, new constraints are added to the classical SVMs formulation in order to improve some performance measurements in the classification. Afterwards, and using the formulation in AMPL, the classification obtained with this approach in Breast Cancer Wisconsin data is compared with the results obtained in Chapter 3.

Chapter 2

Classification Methods

Different classification methods have been proposed in the literature. The differences between them is mainly due to the statistical asspumptions made on the data and the type of algorithms required to build the classifier. The whole set of classification methods can be divided into tree big categories that we will expose right after: linear classifiers, nearest-neighbor classifiers and classification trees, [1].

Before entering in detail in each of the classifiers commented above, we must discuss two important elements in supervised learning: the scoring functions and the performance criteria.

At every moment, we must keep in mind that supervised classification seeks procedures for classifying objects in a set Ω into a set $\mathcal{C}$ of classes. Each object $u_i \in \Omega$ has two components (x_i, y_i), where x_i is called the predictor vector and takes values on a set X such as e.g. $\mathbb{R}^r$ and $y_i \in \mathcal{C}$ is the class membership of u_i.

Scoring functions. For each $c \in \mathcal{C}$, a scoring function $f_c : X \to \mathbb{R}$ is built from the training set $\mathcal{I} \in \Omega$ (objects in Ω whose membership is known), and upcoming objects with predictor vector x are classified in the class $y(x)$ in $\mathcal{C}$ with highest score. That is to say, the classifier is given by

$$y(x) \in \arg\max_{c \in \mathcal{C}} \ f_c(x).$$

In the two-class case, when $\mathcal{C} = \{-1, 1\}$, setting $f_{-1} = 0$, the expression above takes the form

$$y(x) = sign(f_1(x)),$$

where $sign(a) = 1$ if $a > 0$ and $sign(a) = -1$ if $a < 0$.

Performance criteria. Many criteria can be used to evaluate the performance of supervised classification. For instance, the main one in the so-called *generalization power*, that tells how good the built classifier would correctly classify upcoming objects. Another example is the *accuracy*, i.e., the fraction of correctly classified individuals.

Different criteria are needed if the misclassification costs differ between the classes. One of the performance measurement preferred in this case is the so-called area under the Receiver Operating Characteristic (ROC) curve, *AUC*. The *ROC curve* shows the *sensitivity* (proportion of correctly classified objects of class 1) against the *specificity* (proportion of correctly classified objects of class -1).

Sensitivity and Specificity are not more than particular cases of the so-called *Correct Clasification Rate (CCR)*, which gives us the fraction of samples well classified in each class $\in \mathcal{C}$. Other performance metrics are, for example: precision, lift, F-score,..., [2].

2.1 Linear classifiers

One of the simplest classifiers that can be obtained are those in which we assume the scoring functions f_c to be affine, i.e.,

$$f_c(x) = w_c^t x + \beta_c.$$

Hence, the set of samples classified in class c is given by the polyhedron

$$X_c = \{x \in \mathbb{R} : w_c^t x + \beta_c \geq w_{c'}^t + \beta_{c'} \quad \forall c' \in \mathcal{C}\}$$

Another popular linear classifier is obtained in Linear Discriminant Analysis (LDA), summarized as follows: For each class $\in \mathcal{C}$, the predictor vector x of all individuals in such class, is assumed to follow a Gaussian distribution with mean μ_c and common covariance matrix Σ. Also, let π_c be the *prior probability* of an incoming individual of being in group c, given. For a given predictor vector, its posterior probability ($p_c(x)$) of each class is calculated by Bayes theorem, and a sample is classified in its most (posterior) probable class. Then, the scoring function f_c for this classifier is

$$f_c(x) = x^t \Sigma^{-1} \mu_c - \frac{1}{2} \mu_c^t \Sigma^{-1} \mu_c + log(\pi_c).$$

It should be remarked that, in practice, the parameters π_c, μ_c and Σ are unknown but they can be estimated via sample estimates.

And the list of linear classifiers does not stop here: there are many other linear classifiers such as Support Vector Machines, in which our work will focus.

2.2 Nearest-neighbor classifiers

Let δ be a dissimilarity measure in X. In the basic k-nearest neighbors method, for a given sample x, let $N_k(x)$ be the set of k objects in the training sample $\mathcal{I}$ nearest to x, according to δ. Then, x will be classified in the most frequent class c in $N_k(x)$. Hence, the scoring function of the k-nearest neighbors method, $f_c(x)$, is equal to the cardinality of the subset of $N_k(x)$ with label c. Mathematical optimization appears in three different aspects in this method: evaluation of δ, choice of δ and data (prototype) selection.

2.3 Classification trees

A classification tree is a classifier defined as series of *if-then* rules. This classifier is identified with a rooted tree T, in which each node represents a partition of the space X. In the simplest case, T is a binary tree, where each node symbolizes a partition of X into the sets $\{x \in X : x_V \geq t\}$ and $\{x \in X : x_V < t\}$ for some variable V and threshold t. The set of leaf nodes of the tree, S, induces a partition $\{X_s : s \in S\}$ of X and the classifier assigns to any $x \in X_s$ the most frequent class in the training sample whose predictor vector belongs to X_s.

Chapter 3

Support Vector Machines

Support Vector Machines are one of the most popular machine learning algorithms. They became very popular in early 90s and today they are still considered one of the most powerful classification algorithms due to their high performance with very little tunning. In this chapter we overview Support Vector learning, starting with linear SVMs and following with their extension to the nonlinear case, [3].

3.1 Linear Support Vector Machine

In the binary classification setting, assume we have available a non-empty set of data Ω, where each $u_i \in \Omega$ has two components:

$$\Omega = \{u_i = (x_i, y_i) : i = 1, 2, \ldots, n\}$$

where $x_i \in \mathbb{R}^r$ is the so-called predictor vector, and $y_i \in \{1, -1\}$ are two given classes of u_i. We now have a non-empty set $\mathcal{I}$, which will be called the learning set and that will be composed of $u_i = (x_i, y_i)$, where y_i is given, $\forall\ i$. The two-class classification problem is based on predicting, from the data of $\mathcal{I}$, the y_i class of a given $u_i \in \Omega$. It is used $\beta \in \mathbb{R}^r$ and $\beta_0 \in \mathbb{R}$ in order to build a function $f : \mathbb{R}^r \to \mathbb{R}$ such that:

$$f(x) = \beta^t x + \beta_0.$$

This function, called separation function, [7], classifies as class 1 those $x_i \in \mathbb{R}^r$ with $f(x) > 0$ and as class -1 those $x_i \in \mathbb{R}^r$ with $f(x) < 0$.

The goal is to have a function f such that all positive points ($y_i = 1$) in $\mathcal{I}$ are assigned to class 1 and negative points ($y_i = -1$) in $\mathcal{I}$ to class -1. Points x with $f(x) = 0$ will be classified according to a predeterminied rule. According to that,

$$y_i(\beta^t x_i + \beta_0) \geq 0 \qquad \forall i \in \mathcal{I}$$

3.1.1 The Linearly Separable Case

First, we will consider the simplest case, in which we suppose that the positive ($y_i = 1$) and negative ($y_i = -1$) data points from the learning set $\mathcal{I}$ can be separated by a hyperplane of the form

$$\{x : f(x) = \beta_0 + x^t\beta = 0\}$$

where β is the weight vector with norm $\|\beta\|$, and β_0 is the bias. If this hyperplane can separate the two given classes of the learning set without error, the hyperplane is called a *separating hyperplane*.

If positive and negative data points can be separated by the hyperplane $H^0 := \beta_0 + x^t\beta = 0$, then

$$H^+ = \beta_0 + \beta^t x_i > 0, \ if \ y_i = 1$$
$$H^- = \beta_0 + \beta^t x_i < 0, \ if \ y_i = -1$$

For separable sets, there are an infinite number of such hyperplanes. So in order to construct the classifier we have to decide on a reasonable way which of those separating hyperplanes to use. The simplest choice is the maximal margin hyperplane, which can be built as follows: first, consider any separating hyperplane. Then, let d_- and d_+ be, respectively, the shortest distance from this separating hyperplane to the nearest negative data point and to the nearest positive one. We say that the hyperplane is an optimal separating hyperplane if it maximizes the distance between the hyperplane and the closest observation.

In order to find the best separating hyperplane, we use a norm $\|\cdot\|$ in $\mathbb{R}^r$, and derive the distances between the two given classes and the separating hyperplane. First, let us consider the Euclidean case, with the Euclidean norm $\|x\|^2 = x^t x$:

Property. Let $\|\cdot\|$ be the Euclidean distance. Then, given x, we have

$$d_- = d(x, \{y : \beta_0 + \beta^t y \leq 0\}) = \max \frac{\{\beta_0 + \beta^t x, 0\}}{\|\beta\|} \tag{3.1}$$

$$d_+ = d(x, \{y : \beta_0 + \beta^t y \geq 0\}) = \max \frac{\{-(\beta_0 + \beta^t x), 0\}}{\|\beta\|} \tag{3.2}$$

Proof. Let x be a fixed point. We have the following optimization problem, which formulate the distance between the point x and the separating hyperplane:

$$\begin{array}{ll} \min & \|x-y\| \\ \text{subject to:} & \beta_0+\beta^t y=0 \end{array} \tag{3.3}$$

which is equivalent to

$$\begin{array}{ll} \min & \|x-y\|^2 \\ \text{subject to:} & \beta_0+\beta^t y=0 \end{array} \tag{3.4}$$

As we are using the Euclidean distance, we can use the Karush-Kuhn-Tucker (KKT) conditions. Let $\mathcal{L}(y,\lambda)$ be the Lagrange function defined as:

$$\mathcal{L}(y,\lambda)=\|y-x\|^2-\lambda(\beta_0+\beta^t y)$$

Proceeding as the method KKT says:

$$\frac{\partial}{\partial y}\mathcal{L}(y,\lambda)=0: \quad 2(y-x)-\lambda\beta=0$$
$$\frac{\partial}{\partial \lambda}\mathcal{L}(y,\lambda)=0: \quad \beta_0+\beta^t y=0 \Rightarrow \beta^t y=-\beta_0.$$

Thus, if we multiply $\dfrac{\partial}{\partial y}\mathcal{L}(y,\lambda)$ on the left by β^t and operate

$$2\beta^t(y-x)-\lambda\beta^t\beta=0 \underset{\beta^t y=-\beta_0}{\Rightarrow} -2\beta_0-2\beta^t x=\lambda\beta^t\beta.$$

And if we solve for λ, we get

$$\lambda=\frac{-2\beta_0-2\beta^t x}{\beta^t\beta}.$$

Applying norm $\|\cdot\|$ and replacing λ in equation of $\dfrac{\partial}{\partial y}\mathcal{L}(y,\lambda)$ we obtain

$$2(y-x)=\lambda\beta \underset{\text{App. norm } \|\cdot\|}{\Longrightarrow} \|y-x\|=\frac{|\lambda|}{2}\|\beta\| \underset{\text{Repl. } \lambda}{=} \left|\frac{-2\beta_0-2\beta^t x}{\beta^t\beta}\right|\frac{\|\beta\|}{2}=$$
$$=\frac{|\beta_0+\beta^t x|}{\|\beta\|^2}\|\beta\|=\frac{|\beta_0+\beta^t x|}{\|\beta\|}.$$

Summarizing, in the Euclidean case:

$$d(x, \{y : \beta_0 + \beta^t y = 0\}) = \frac{|\beta_0 + \beta^t x|}{\|\beta\|}$$

If $\beta_0 + \beta^t x \geq 0$,

$$d(x, \{y : \beta_0 + \beta^t y \geq 0\}) = 0$$

$$d(x, \{y : \beta_0 + \beta^t y \leq 0\}) = \frac{|\beta_0 + \beta^t x|}{\|\beta\|} = \frac{\beta_0 + \beta^t x}{\|\beta\|}$$

If $\beta_0 + \beta^t x \leq 0$,

$$d(x, \{y : \beta_0 + \beta^t y \leq 0\}) = 0$$

$$d(x, \{y : \beta_0 + \beta^t y \geq 0\}) = \frac{|\beta_0 + \beta^t x|}{\|\beta\|} = \frac{-(\beta_0 + \beta^t x)}{\|\beta\|}$$

In general:

$$d(x, \{y : \beta_0 + \beta^t y \geq 0\}) = \max\left\{0, \frac{-(\beta_0 + \beta^t x)}{\|\beta\|}\right\}$$

$$d(x, \{y : \beta_0 + \beta^t y \leq 0\}) = \max\left\{0, \frac{\beta_0 + \beta^t x}{\|\beta\|}\right\} \qquad \square$$

For another arbitrary norm, we can use the next result, which extends the former property, [10]:

Theorem 1.1. For any norm $\|\cdot\|$ and any hyperplane $H(\beta, \beta_0)$ we have

$$d_{\|\cdot\|}(x, H(\beta, \beta_0)) = \begin{cases} \dfrac{\beta_0 - \beta^t x}{\|\beta\|^\circ}, & when\ \beta^t x \leq \beta_0, \\ \\ \dfrac{\beta^t x - \beta_0}{\|\beta\|^\circ}, & when\ \beta^t x > \beta_0. \end{cases}$$

Here $\|\beta\|^\circ$ denotes the dual norm of $\|\beta\|$, which is defined as

$$\begin{array}{rcll} \|\beta\|^\circ & = & \max & u^t\beta \\ & & \text{subject to:} & \|u\| = 1 \end{array}$$

In the theorem 3.1.1, we have the formula of the distance from one point x to a half-space. Now, given $(x_1, \ldots, x_n)$ with labels $(y_1, \ldots, y_n)$, the distance of x_i to the halfspace of misclassification is given by

$$d_i = \frac{\max\{y_i(\beta_0 + \beta^t x_i), 0\}}{\|\beta\|^0}, \quad \forall\, i \in \mathcal{I}.$$

The minimum of this equation, $d_{\mathcal{I}}=\min_{u_i \in \mathcal{I}}\ d_i$, is called margin.

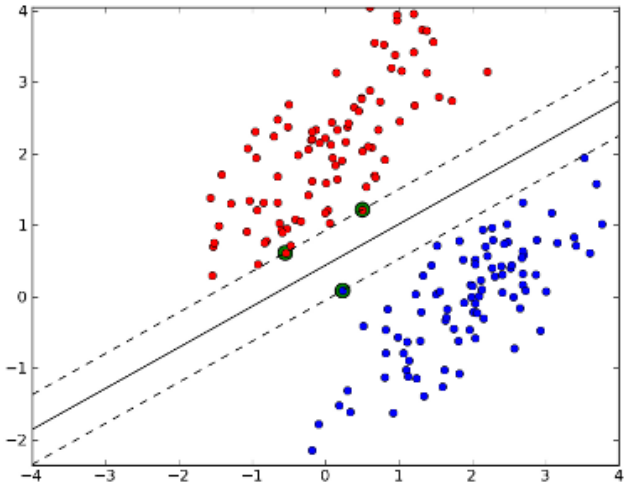


Figure 3.1: Linear SVM with the margin

The goal is to maximize the margin. We get that solving the following optimization problem:

$$\max_{\beta,\beta_0} \quad \min_i \quad \frac{\max\{y_i(\beta_0 + \beta^t x_i), 0\}}{\|\beta\|^\circ},$$

which is equivalent to,

$$\min_{\beta,\beta_0} \quad \max_i \frac{\|\beta\|^\circ}{\max\{y_i(\beta_0 + \beta^t x_i), 0\}},$$

or,

$$\min_{\beta,\beta_0} \quad \frac{\|\beta\|^\circ}{\min_i \max\{y_i(\beta_0 + \beta^t x_i), 0\}}.$$

The function $(\beta_0, \beta) \longmapsto \dfrac{\|\beta\|^\circ}{\min_i \max\{y_i(\beta_0 + \beta^t x_i), 0\}}$ is homogeneus in $\mathbb{R}_+$ and hence we can assume (without loss of generality) that the denominator equals 1. Thus, we have the next equivalent representation:

$$\begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|^\circ \\ \text{subject to:} & \min_i \quad y_i(\beta_0 + \beta^t x_i) = 1 \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{3.5}$$

which is equivalent to,

$$\begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|^\circ \\ \text{subject to:} & \min_i \quad y_i(\beta_0 + \beta^t x_i) \geq 1 \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{3.6}$$

and equivalent to,

$$\begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|^\circ \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \ \forall \ i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{3.7}$$

In the Euclidean case we have

$$\begin{array}{ll} \min_{\beta_0,\beta} & \beta^t\beta \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) = 1, \ \forall \ i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{3.8}$$

which is an optimization problem with convex objective function and linear constraints. Then, the problem above can be equivalent to:

$$\begin{array}{ll} \min_{\beta_0,\beta} & \beta^t\beta \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \ \forall \ i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{3.9}$$

For polyhedral norms, problem (3.7) can be written as a linear problem. Let us consider the particular important cases $\|\cdot\| = \|\cdot\|_1$ and $\|\cdot\| = \|\cdot\|_\infty$. To achieve the dual of those norms we have the following property:

Property. Let $\|\cdot\|_p$ be a $p - norm$. Then, its dual norm is $\|\cdot\|_p^\circ = \|\cdot\|_q$, where p and q satisfy

$$\frac{1}{p} + \frac{1}{q} = 1.$$

If we have $\|\cdot\| = \|\cdot\|_1$, then its dual $\|\cdot\|^\circ$ is the infinity norm $\|\cdot\|_\infty$, and the problem (3.7) can be expressed as follows:

$$\begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|_\infty \\ \text{subject to:} & \min_i \ \ y_i(\beta_0 + \beta^t x_i) \geq 1, \ \forall \ i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{3.10}$$

This problem can be reformulated as a linear problem,

$$\begin{array}{ll} \min & z \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \ \forall \ i \in \mathcal{I} \\ & z \geq \beta_i \geq -z \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R}, z \geq 0 \end{array} \tag{3.11}$$

On the other hand, if we have $\|\cdot\| = \|\cdot\|_\infty$, then its dual $\|\cdot\|^\circ$ is the 1-norm $\|\cdot\|_1$, and our problem can be expressed as follows:

$$\begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|_1 \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \ \forall \ i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{3.12}$$

which can also be converted in a linear problem,

$$\begin{array}{ll} \min & \sum_j z_j \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \ \forall \ i \in \mathcal{I} \\ & z_j \geq \beta_i \geq -z_j, \ j = 1, \ldots, r \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R}, z_j \geq 0 \end{array} \tag{3.13}$$

3.1.2 The Linearly Nonseparable Case

The previous section discussed the case where it is possible to linearly separate the data that belong to different classes. As our intuition says, the previous formulation will not find a solution if the data cannot be separated by a hyperplane. In fact, in practical applications for real data, it is unlikely that there will be such a clear linear separation between the two classes. More likely, there will be some overlap.

The overlap will cause problems for any classification rule, and depending upon the extent of the overlap, the overlapping points could not be classified.

The nonseparable case occurs if either the two classes are separable, but not linearly, or if no clear separability exists between the two classes, linearly or nonlinearly.

As we mentioned before, in the previous section we assumed that $\mathcal{I}$ was linearly separable. If this is not the case, the optimization problem formulated above is infeasible. Therefore, we must find other method.

One of such method to solve the nonseparable case is to create a more flexible formulation of the problem, which leads to a soft-margin solution through maximization of this soft-margin. Let ε be a perturbation. Starting from an infeasible problem, it is possible to perturb the constraints in order to make it feasible. We must introduce a penalty in the objective function to control the perturbation. Hence, the following problem can be formulated as

$$\begin{array}{ll} \min & \beta^t\beta + C\|\varepsilon\|_r^r \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1, \forall\ i \in \mathcal{I}. \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall i \in \mathcal{I} \end{array} \tag{3.14}$$

where $C > 0$ is the so-called *regularization parameter*. As an example, we can discuss the case of ℓ_1 regularization. Then, the previous problem can be written as

$$\begin{array}{ll} \min & \beta^t\beta + C\sum_{i\in\mathcal{I}} \varepsilon_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1, \forall\ i \in \mathcal{I}. \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall i \in \mathcal{I} \end{array} \tag{3.15}$$

equivalent to,

$$\begin{array}{ll} \min & \frac{1}{2}\beta^t\beta + C\sum_{i\in\mathcal{I}} \varepsilon_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1, \forall\ i \in \mathcal{I}. \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall i \in \mathcal{I} \end{array} \tag{3.16}$$

In the Euclidean case, we apply again the KKT method. So, fixed $\lambda = (\lambda_1, \ldots, \lambda_n)^t \geq 0$ and $\eta = (\eta_1, \ldots, \eta_n)^t \geq 0$, let $\mathcal{L}(\beta,\beta_0, \lambda, \eta\ ,\varepsilon_i)$ be the Lagrange function, defined as

$$\mathcal{L}(\beta, \beta_0, \lambda, \eta, \varepsilon_i) =$$

$$= \frac{1}{2}\beta^t\beta + C\sum_{i\in\mathcal{I}} \varepsilon_i - \sum_{i\in\mathcal{I}} \lambda_i[y_i(\beta_0 + \beta^t x_i) - (1 - \varepsilon_i)] - \sum_{i\in\mathcal{I}} \eta_i\varepsilon_i$$

Then, proceeding as the KKT method says,

$$\frac{\partial}{\partial\beta}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\,\varepsilon_i)=0:\quad \beta-\textstyle\sum_{i\in\mathcal{I}}\lambda_i y_i x_i=0$$
$$\frac{\partial}{\partial\beta_0}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\,\varepsilon_i)=0:\quad -\textstyle\sum_{i\in\mathcal{I}}\lambda_i y_i=0$$
$$\frac{\partial}{\partial\varepsilon_i}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\,\varepsilon_i)=0:\quad C-\lambda_i-\eta_i$$

So from $\dfrac{\partial}{\partial\beta}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\,\varepsilon_i)=0$ we obtain

$$\beta=\textstyle\sum_{i\in\mathcal{I}}\lambda_i y_i x_i$$

And from $\dfrac{\partial}{\partial\varepsilon_i}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\,\varepsilon_i)=0$ we get

$$\lambda_i=C-\eta_i.$$

Substituting in the equation $\mathcal{L}(\beta,\beta_0,\lambda,\eta,\varepsilon_i)$ we get the dual of the problem formulated above. Thus,

$$\begin{array}{ll}\max & \sum_{i\in\mathcal{I}}\lambda_i-\frac{1}{2}\sum_{i,j\in\mathcal{I}}\lambda_i\lambda_j y_i y_j x_i^t x_j\\ \text{subject to:} & \sum_{i\in\mathcal{I}}y_i\lambda_i=0\\ & \lambda_i=C-\eta_i,\quad\forall\, i\in\mathcal{I}\\ & \lambda_i\geq 0,\quad\forall\, i\in\mathcal{I}\\ & \eta_i\geq 0,\quad\forall\, i\in\mathcal{I}\end{array}\tag{3.17}$$

and this can be rewritten as

$$\begin{array}{ll}\max & \sum_{i\in\mathcal{I}}\lambda_i-\frac{1}{2}\sum_{i,j\in\mathcal{I}}\lambda_i\lambda_j y_i y_j x_i^t x_j\\ \text{subject to:} & \sum_{i\in\mathcal{I}}y_i\lambda_i=0\\ & 0\leq\lambda_i\leq C,\quad\forall\, i\in\mathcal{I}\end{array}\tag{3.18}$$

We can observe that $0\leq\lambda_i\leq C$. Also, we get $\lambda_i=C$ when $\varepsilon_i>0$ (which is the case when $y_i(\beta_0+\beta^t x_i)<1$). In addition, when $y_i(\beta_0+\beta^t x_i)>1$, $\varepsilon_i=0$ since no cost is incurred, and thus $\lambda_i=0$. Futhermore, when $y_i(\beta_0+\beta^t x_i)=1$, λ_i can lie between 0 and C, [5].

3.2 Nonlinear Support Vector Machine

Up to now, we have only discussed methods for constructing linear SVM classifiers. But there are many cases where linear classifiers are not appropriate for the data learning set. An extension to nonlinear decision surfaces

is necessary since real-life classification problems are hard to be solved by a linear classifier.

The main key to constructing a nonlinear SVM is to observe that the observations in Ω only enter the dual optimization problem through the inner products $\langle x_i, x_j \rangle = x_i^t x_j,\ i, j = 1, 2, \ldots, n$. In order to build a nonlinear SVM, we need some nonlinear tranformations, [7]. Let ϕ be a nonlinear map, called the $feature\ map$, and let $\mathcal{H}$ be an $N_{\mathcal{H}}$-dimensional $feature\ space$. The space $\mathcal{H}$ may be very high-dimensional, possibly even infinite-dimensional. We will generally assume that $\mathcal{H}$ is a Hilbert space of real-valued functions on $\mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$.

Suppose we transform each observation, $x_i \in \mathbb{R}^r$, in Ω using some nonlinear mapping $\phi : \mathbb{R}^r \to \mathcal{H}$. Hence,

$$\phi(x_i) = (\phi_1(x_i), \ldots, \phi_{N_{\mathcal{H}}}(x_i))^t \in \mathcal{H},\ \forall\ i = 1, 2, \ldots, n.$$

The transformed sample is then $(\phi(x_i), y_i)$, where $y_i \in \{-1, 1\}$ identifies the two classes. In this new space we must work with the learning set of data $\hat{\mathcal{I}}$=$\{(\phi(x_i), y_i) : \forall\ u_i \in \mathcal{I}\}$, which is linearly separable. If we substitute $\phi(x_i)$ by x_i in the development of the linear SVM, then the data would only enter the optimization problem by way of the inner products $\langle \phi(x_i), \phi(x_j) \rangle = \phi(x_i)^t \phi(x_j)$.

The dificulty in using nonlinear transformations in this way is computing such inner products in high-dimensional space $\mathcal{H}$.

Now, we want to solve the nonlinear problem. We proceed as in the linearly separable case and seek $\beta \in \mathcal{F}$ and $\beta_0 \in \mathbb{R}$ to classify data points according to the rule f:

$$f(x) = \beta^t \phi(x) + \beta_0$$

Rule f is linear on the data when we transform it with the mapping ϕ, but f is not linear on the original space $\mathbb{R}^r$. Rule f assignes x to class 1 if $f(x) > 0$ and to class -1 if $f(x) < 0$.

The problem of maximizing the margin mentioned above, can be rewritten as

$$\begin{array}{ll} \min & \|\beta\|^\circ \\ \text{subject to:} & y_i(\beta_0 + \beta^t \phi(x_i)) \geq 1, \forall\ i \in \mathcal{I} \\ & \beta \in \mathcal{F}, \beta_0 \in \mathbb{R} \end{array} \tag{3.19}$$

If we use the Euclidean norm to measure distances in the new transformed space, our previous problem can be written as:

$$\begin{array}{ll} \min & \beta^t\beta \\ \text{subject to:} & y_i(\beta_0 + \beta^t\phi(x_i)) \geq 1, \forall\ i \in \mathcal{I} \\ & \beta \in \mathcal{F}, \beta_0 \in \mathbb{R}. \end{array} \tag{3.20}$$

This problem is equivalent to,

$$\begin{array}{ll} \min & \frac{1}{2}\beta^t\beta \\ \text{subject to:} & y_i(\beta_0 + \beta^t\phi(x_i)) \geq 1, \forall\ i \in \mathcal{I} \\ & \beta \in \mathcal{F}, \beta_0 \in \mathbb{R} \end{array} \tag{3.21}$$

We now build its dual. In order to do this, we need to use the KKT method. Fixed $\lambda = (\lambda_1, \ldots, \lambda_n)^t \geq 0$, let $\mathcal{L}(\beta, \lambda)$ be the Lagrange function as follows

$$\mathcal{L}(\beta, \beta_0, \lambda) = \frac{1}{2}\beta^t\beta - \textstyle\sum_{i\in\mathcal{I}} \lambda_i[y_i(\beta_0 + \beta^t\phi(x_i)) - 1]$$

And proceeding as the KKT method says,

$$\frac{\partial}{\partial\beta}\mathcal{L}(\beta, \beta_0, \lambda) = 0: \quad \beta - \textstyle\sum_{i\in\mathcal{I}} \lambda_i y_i \phi(x_i) = 0$$
$$\frac{\partial}{\partial\beta_0}\mathcal{L}(\beta, \beta_0, \lambda) = 0: \quad -\textstyle\sum_{i\in\mathcal{I}} \lambda_i y_i = 0$$

Thus, by $\dfrac{\partial}{\partial\beta}\mathcal{L}(\beta, \beta_0, \lambda) = 0$ we have

$$\beta = \textstyle\sum_{i\in\mathcal{I}} \lambda_i y_i \phi(x_i)$$

Replacing results in $\mathcal{L}(\beta, \beta_0, \lambda)$ we have its dual,

$$\begin{array}{ll} \max & \textstyle\sum_{i\in\mathcal{I}} \lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}} \lambda_i\lambda_j y_i y_j \phi(x_i)^t\phi(x_j) \\ \text{subject to:} & \textstyle\sum_{i\in\mathcal{I}} y_i\lambda_i = 0 \\ & \lambda_i \geq 0,\ \forall\ i \in \mathcal{I} \end{array} \tag{3.22}$$

3.2.1 The "Kernel Trick"

We should remember that the idea behind nonlinear SVM was to find an optimal separating hyperplane in the high-dimensional feature space $\mathcal{H}$ just as we did for the linear SVM in the input space. The optimal separating hyperplane can be formulated with or without slack variables, as appropriate.

At first, someone could expect that the dimensionality of $\mathcal{H}$ is a huge problem to build an optimal separating hyperplane and a classification rule, because of the course of dimensionality. But thanks to the "Kernel Trick", which was first applied to SVM by Cortes and Vapnik in 1995, [4], we do not have to calculate the inner products explicitly, so dimensionality is not a problem.

The so-called kernel trick is an idea that is widely used in algorithms for computing inner products of the form $\langle \phi(x_i), \phi(x_j) \rangle$ in feature space $\mathcal{H}$. The trick is that instead of computing these inner products in $\mathcal{H}$, which would be computationally expensive because of its high dimensionality, we compute using a nonlinear kernel function, $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, in the input space, which helps us to speed up the computations. With this support, we just compute a linear SVM, but where the computations are carried out in some other space.

3.2.2 Kernels and Their Properties

A kernel K is a function $K : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$, given by

$$K(x_i, z_i) = \langle \phi(x_i), \phi(z_i) \rangle, \ \forall \ x_i, z_i \in \mathbb{R}^r.$$

The kernel function is designed to compute inner-products in $\mathcal{H}$ by using only the original input data. Thus, wherever we see the inner product $\langle \phi(x_i), \phi(z_i) \rangle$, we substitute the kernel function $K(x_i, z_i)$. The choice of K implicitly determines both ϕ and $\mathcal{H}$. As example, the problem (3.22) can be reformulated using the "Kernel Trick" as

$$\begin{array}{ll} \max & \sum_{i \in \mathcal{I}} \lambda_i - \dfrac{1}{2} \sum_{i,j \in \mathcal{I}} \lambda_i \lambda_j y_i y_j K(x_i, x_j) \\ \text{subject to:} & \sum_{i \in \mathcal{I}} y_i \lambda_i = 0 \\ & \lambda_i \geq 0, \quad \forall \ i \in \mathcal{I} \end{array} \tag{3.23}$$

The big advantage of using kernels as inner products is that, if we are given a kernel function K, then we do not need to know the explicit form of ϕ.

Kernel properties

Some kernels properties are

1. If $K_1(x_i, y_i)$ is a kernel and we have a positive real number a_1, then

$$K(x_i, z_i) = a_1 K_1(x_i, z_i)$$

is a kernel.

2. If $K_1(x_i, z_i)$ and $K_2(x_i, z_i)$ are two kernels and a_1, a_2 are two positive real numbers, then

$$K(x_i, z_i) = a_1 K_1(x_i, z_i) + a_2 K_2(x_i, z_i)$$

is a kernel.

3. The multiplication of two kernels K_1 and K_2 yields a kernel

$$K(x_i, z_i) = K_1(x_i, z_i) K_2(x_i, z_i)$$

4. The above properties imply that any polynomial with possitive coefficients, $pol^+ = \{\sum_{i=1}^{n} \alpha_i x^i \mid n \in \mathbb{N},\ \alpha_1, \ldots, \alpha_n \in \mathbb{R}^+\}$, evaluated at a kernel K_1, yields a kernel

$$K(x_i, z_i) = pol^+(K_1(x_i, z_i))$$

In particular, we have that

$$K(x_i, z_i) = exp(K_1(x_i, z_i))$$

is a kernel by taking the limit of the series expansion of the exponencial function.

5. If g is a real-valued function on $\mathbb{R}^r$, then

$$K(x_i, z_i) = g(x_i) g(z_i)$$

is a kernel.

6. If ψ is a $\mathbb{R}^p$-valued function on $\mathbb{R}^r$, $p < r$, and K_1 is a kernel on $\mathbb{R}^p \times \mathbb{R}^p$, then

$$K(x_i, z_i) = K_1(\psi(x_i), \psi(z_i))$$

is also a kernel.

7. If A is a positive definite matrix of size $r \times r$, then

$$K(x_i, z_i) = (x_i)^t A z_i$$

is a kernel.

We require that the kernel function be symmetric, $K(x_i, z_i) = K(z_i, x_i)$, and satisfy the inequality, $[K(x_i, z_i)]^2 \leq K(x_i, x_i)K(z_i, z_i)$, derived from the Cauchy-Schwarz inequality. If $K(x_i, x_i) = 1 \ \forall \ x_i \in \mathbb{R}^r$, this implies that $\|\phi(x_i)\|_{\mathcal{H}} = 1$. A kernel K is said to have the reproducing property if, for any $f \in \mathcal{H}$,

$$\langle f(\cdot), K(x_i, \cdot)\rangle = f(x_i)$$

If K has this property, we say it is a reproducing kernel. K is also called the representer of evaluation. In particular, if $f(\cdot) = K(\cdot, x_i)$, then,

$$\langle K(x_i, \cdot), K(z_i, \cdot)\rangle = K(x_i, z_i)$$

Let $\{x_1, \ldots, x_n\}$ be any set of n points in $\mathbb{R}^r$. Then, the $(n \times n)$-matrix $\mathbf{K} = (K_{ij})$, where $(K_{ij}) = K(x_i, x_j),\ i, j = 1, 2, \ldots, n$, is called the *Gram* matrix of K with respect to $\{x_1, \ldots, x_n\}$. If the *Gram* matrix $\mathbf{K}$ satisfies $u^t \mathbf{K} u \geq 0$, for any n-vector u, then it is said to be nonnegative-definite with nonnegative eigen values, in which case we say that K is a nonnegative-definite kernel or *Mercer kernel*.

If K is a specific *Mercer kernel* on $\mathbb{R}^r \times \mathbb{R}^r$, we can always construct a unique Hilbert space $\mathcal{H}_{\mathcal{K}}$, say, of real-valued functions for which K is its reproducing kernel. We call $\mathcal{H}_{\mathcal{K}}$ a real *reproducing kernel Hilbert space* (rkhs). We can write the inner product and norm of $\mathcal{H}_{\mathcal{K}}$ by $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\mathcal{K}}}$ and $\| \cdot \|_{\mathcal{H}_{\mathcal{K}}}$, respectively.

A kernel is called stationary, or traslation-invariant, if it has the general form $K(x_i, z_i) = k(x_i - z_i)$, where $k : \mathbb{R}^r \longrightarrow \mathbb{R}$. A kernel $K(x_i, z_i)$ is isotropic if it depends only upon the distance $\delta = \|x_i - z_i\|$, i.e., if $K(x_i, z_i) = k(\delta)$, scaled to have $k(0) = 1$.

3.2.3 Examples of Kernels

We have the following table with some kernel functions, $K(x_i, z_i)$, where $\sigma > 0$ is a scale parameter, $a, b, c \geq 0$ and d is an integer. We are also using the Euclidean norm $\|x\|^2 = x^t x$.

Kernel	$\boldsymbol{K(x_i, z_i)}$
Polynomial of degree d	$(x_i^t z_i + c)^d$
Radial basis function	$\exp\left(-\frac{\|x_i - z_i\|^2}{2\sigma^2}\right)$
Laplacian	$\exp\left(\frac{\|x_i - z_i\|}{\sigma}\right)$
Thin-plate spline	$\left(\frac{\|x_i - z_i\|}{\sigma}\right)^2 \log\left(\frac{\|x_i - z_i\|}{\sigma}\right)$
Sigmoid	$\tanh(ax_i^t z_i + b)$

Table 3.1: Examples of Kernel functions

As an example of these kernels we have the inhomogeneous polynomial kernel of degree d,

$$(x_i^t z_i + c)^d, \quad x_i, z_i \in \mathbb{R}^r$$

where c and d are parameters. The homogeneous form of the kernel occurs when $c = 0$ in the expression above. If $d = 1$ and $c = 0$, the feature map reduces to the identity. Usually, we take $c > 0$.

A simple nonlinear map is given by the case $\mathbb{R}^r = \mathbb{R}^2$ and $d = 2$. If $x_i = (x_1, x_2)^t$ and $z_i = (z_1, z_2)^t$, then,

$$K(x_i, z_i) = (x_i^t z_i + c)^2 = (x_1 z_1 + x_2 z_2 + c)^2 = \langle \phi(x_i), \phi(z_i) \rangle$$

where $\phi(x_i) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2c}x_1, \sqrt{2}x_2, c)^t$ and similary for $\phi(z_i)$. In this example, the function $\phi(x_i)$ consists of six features ($\mathcal{H}= \mathbb{R}^6$), all monomials having degree at most 2. For this kernel, we see that c controls the magnitudes of the constant term and the first-degree term.

In general, there will be $dim(\mathcal{H}) = \binom{r+d}{d}$ different features, consisting of all monomials having degree at most d.

Other popular kernels are given in Table 3.1. For example, the radial basis

function, Laplacian, and thin-plate spline kernels are example or stationary, or translation-invariant, kernels having the general form $K(x_i, z_i) = k(x_i - z_i)$, as we explained before. The polynomial kernel is an example of a nonstationary kernel.

Strictly speaking, the Sigmoid kernel is not a kernel. It satifies Mercer's conditions only for certain values of a and b. Despite this, it has become very popular in that role in certain situations (e.g., two-layer neural networks).

It is not always obvious which kernel to choose in any given application. Prior knowledge or a search through the literature can be helpful. If no such information is available, the most popular approach is to try either a radial basis function, which has only a single parameter, σ, to be determined, or a polynomial kernel of low degree($d = 1$ or 2). If it is necessary, more complicated kernels can then applied to compare results.

3.3 Multi-class SVM

As studied before, Support Vector Machines were initially designed for binary classification. The traditional way to perform multi-class classification is to use one of the following methods:

- one-against-one classification,
- one-against-all classification, or
- DAGSVM.

In the following subsections we will focus only in the two first: one-against-one and one-against-all classifications. The third approach can be seen more in detail in [11].

3.3.1 One-against-one

In this approach, suppose we have available a non-empty set of data Ω, where each $u_i \in \Omega$ has two components:

$$\Omega = \{u_i = (x_i, y_i) : i = 1, \ldots, n\}$$

with $x_i \in \mathbb{R}^r$ an r-dimensional vector of predictor variables and $y_i \in \{1, 2, \ldots, M\}$, M given classes of u_i.

As before in *Section 3.1*, we have the non-empty learning set $\mathcal{I}$, composed again of $u_i = (x_i, y_i)$, where y_i is given, $\forall i$. The multi-class classification problem is based on predicting, from the data of $\mathcal{I}$, the y_i class of a given $u_i \in \Omega$.

A well-known method is the so-called "one-against-one method". This method builds $M(M-1)/2$ binary classifiers that are trained to distinguish the samples of one class from the samples of another class. For training data from the kth and the lth classes, the following binary classification problem is solved:

$$\begin{array}{ll} \min & \sum_{i\in\mathcal{I}:y_i=k,l} \lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}:y_i=k,l;y_j=k,l} \lambda_i\lambda_j y_i y_j \phi(x_i)'\phi(x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}:y_i=k,l} y_i\lambda_i = 0 \\ & 0 \le \lambda_i \le C, \forall i \in \mathcal{I} : y_i = k, l \end{array}$$

There are different procedures for doing the future testing after all $M(M-1)/2$ classifiers are constructed. For example, the following voting strategy, called "Max Wins": if $sign(\beta_0 + \beta^t\phi(x))$ (where $\beta_0 = \frac{1}{N_S}\sum_{s\in S}(y_s - \sum_{m\in S}\lambda_m y_m \phi(x_m)\phi(x_s))$ and $\beta = \sum_{i\in\mathcal{I}:y_i=k,l}\lambda_i y_i \phi(x)$, and S is the set composed by $\{i : \lambda_i \neq 0\}$) says x is in the ith class, then, the vote for the ith class is added by one. Otherwise, the jth is increased by one. Then, we predict x is in the class with the largest vote, [6].

3.3.2 One-against-all

As before, M classes and n samples are considered:

$$\Omega = \{u_i = (x_i, y_i) : i = 1, \ldots, n\},$$

where $x_i \in \mathbb{R}^r$ is a r-dimensional vector of predictor variables and $y_i \in \{1, 2, \ldots, M\}$ is the corresponding class label.

In this way, "one-against-all" approach builds M binary SVM classifiers, where each of the M classifiers is formed by one of the M different classes and the rest, and hence renaming the class labels as detailed in the next paragraph.

Each of the M SVM is trained as follows: For the ith SVM, take all the training samples of the ith class with positive labels (+1), and all the others with negative labels (-1). Hence, we can consider a new set Ω' such that

$$\Omega' = \{u'_i = (x_i, y'_i) : i = 1, \ldots, n\},$$

where $x_i \in \mathbb{R}^r$ is the former r-dimensional vector of predictor variables and $y_i' \in \{-1, +1\}$ is the corresponding new class label after the rename.

Then, this ith SVM solves the following problem that yields the ith decision function $f_i(x) = \beta_i^t \phi(x) + \beta_{0i}$:

$$\begin{array}{ll} \min & \sum_{i \in \mathcal{I}'} \lambda_i - \frac{1}{2} \sum_{i,j \in \mathcal{I}'} \lambda_i \lambda_j y_i' y_j' \phi(x_i)' \phi(x_j) \\ \text{subject to:} & \sum_{i \in \mathcal{I}'} y_i' \lambda_i = 0 \\ & 0 \leq \lambda_i \leq C, \forall i \in \mathcal{I}' \end{array}$$

Finally, a sample x is classified in class i^* so that, [6],

$$i^* = \max_{i=1,\ldots,M} f_i(x) = \max_{i=1,\ldots,M} (\beta_i^t \phi(x) + \beta_{0i}).$$

3.4 SVM in R

Here, we describe some R libraries which allow us to perform SVM in R. A first library we should mention is the library "e1071". This library has three different commands: the first of them, *svm*, is used to build the SVM classifier, the second one, *predict*, predicts values based upon a model trained by "svm", and the third one, "plot.svm", enables us to visualize the SVM results.

Along this section, we will work with the Breast Cancer Wisconsin dataset, [9]. This dataset has 569 samples and 30 real-valued input features, appart from the ID (ID number) and the diagnosis. Diagnosis has two classes: B (bening) or M (malign).

Actually, only ten real-valued features are computed for each cell nucleus:

1) radius (mean of distances from center to points on the perimeter)

2) texture (standard deviation of gray-scale values)

3) perimeter

4) area

5) smoothness (local variation in radius lengths)

6) compactness (perimeter^2 / area - 1)

7) concavity (severity of concave portions of the contour)

8) concave points (number of concave portions of the contour)

9) symmetry

10) fractal dimension ("coastline approximation" - 1)

then, the mean, standard error, and "worst" or largest velue (mean of the three largest values) of these features were computed for each image, resulting in the complete set of 30 features. For instance, field 3 is *Mean Radius*, field 13 is *Radius SE* and field 23 is *Worst Radius*. This dataset does not have missing attribute values and its class distribution is: 357 benign and 212 malign.

In what follows, we are going to see and explain how to use the three different commands in "e1071" mentioned avobe.

Build the SVM classifier.

The command used to build the classifier is called *svm*. Now, we are going to see the use of such command.

If one goes to the R console and writes "`?svm`", it will appear the help window. In the *Usage* part we can see

```
## S3 method for class 'formula'
svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)

## Default S3 method:
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon
= 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,
..., subset, na.action = na.omit)
```

"svm" command is used to train a support vector machine and thus, build the model. It can be used to carry out general regression and classification, as well as density-estimation. A formula interface is provided. If the predictor variable include factors, the formula interface must be used to get a correct

model matrix. The probability model for classification fits a logistic distribution using maximum likelihood to the decision values of all binary classifiers, and computes the a-posteriori class probabilities for the multiclass problem using quadratic optimization. The probabilistic regression model assumes laplace-distributed errors for the predictors, and estimates the scale parameter using maximum likelihood.

Once we have built an SVM clasifier, we have an object containing the fitted model, and including

- `SV`: the resulting support vectors
- `index`: the index of the resulting support vectors in the data matrix. Note that this index refers to the preprocessed data
- `coefs`: the corresponding coefficients times the training labels
- `rho`: the negative intercept
- `sigma`: in case of a probabilistic regression model, the scale parameter of the hypothesized laplace distribution estimated by maximum likelihood
- `probA`, `probB`: numeric vectors of length $\frac{k(k-1)}{2}$, where k is the number of classes, containing the parameters of the logistic distributions fitted to the decision values of the binary classifiers $\left(\frac{1}{(1+exp(ax+b)}\right)$

Now, we can see an example with the Breast Cancer Wisconsin dataset. First, we install and load the "e1071" package:

```
install.packages('e1071', dependencies=TRUE)
library(e1071)
```

After that, we download the dataset and delete its first column, which indicates the patient's ID number:

```
dataset <- read.csv('http://archive.ics.uci.edu/ml/
          machine-learning-databases/breast-cancer-wisconsin/
          wdbc.data', head=FALSE)
datos <- dataset[,2:32]
```

Then, we define index as the *datos*' number of rows. Afterwards, we randomly divide the information between testset and trainset. Test set and train set have, respectively, the 30 % and the 70 % of the samples.

```
index <- 1:nrow(datos)
testindex <- sample(index, trunc(length(index)*30/100))
testset <- datos[testindex,]
trainset <- datos[-testindex,]
```

Now, we can use a function called *tune.svm*. This function tunes the hyperparameters of the svm method using a grid search over supplied parameter ranges:

```
tuned <- tune.svm(V2∽., data = trainset, gamma = 10^(-6:-1),
          cost = 10^(-1:1))
summary(tuned)
```

whose output is:

```
tuning of ‘svm’:

- sampling method:  10-fold cross validation

- best parameters:
          gamma   cost
          0.001   10

- best performance:  0.02269231

- Detailed performance results:
```

```
    gamma  cost  error      dispersion
1   1e-06  0.1   0.38064103 0.06718161
2   1e-05  0.1   0.38064103 0.06718161
3   1e-04  0.1   0.38064103 0.06718161
4   1e-03  0.1   0.28801282 0.05210964
5   1e-02  0.1   0.06025641 0.04768070
6   1e-01  0.1   0.07275641 0.04340948
7   1e-06  1.0   0.38064103 0.06718161
8   1e-05  1.0   0.38064103 0.06718161
9   1e-04  1.0   0.27051282 0.04288968
10  1e-03  1.0   0.05275641 0.04354054
11  1e-02  1.0   0.02769231 0.02527300
12  1e-01  1.0   0.05525641 0.04857604
13  1e-06  10.0  0.38064103 0.06718161
14  1e-05  10.0  0.27051282 0.04288968
15  1e-04  10.0  0.05525641 0.04407908
16  1e-03  10.0  0.03769231 0.03198968
17  1e-02  10.0  0.02269231 0.02240474
18  1e-01  10.0  0.06275641 0.04773855
```

In addition, the names of the variables can be seen typing `names(datos)` in the R console,

```
[1] "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9" "V10" "V11" "V12" "V13"
[13] "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23"
[25] "V24" "V25" "V26" "V27" "V28" "V29" "V30" "V31" "V32"
```

As we mentioned before, variable V1 is the ID number which was previously deleted, so this is the reason of why `"V1"` does not appear in the output above. To conclude, we implement the svm model with the radial basis function kernel, using the best $\gamma = \dfrac{1}{2\sigma^2}$ and C parameters according to the results obtained in the "tune.svm" output, i.e. $\gamma = 0.001$ and $C = 10$:

```
model <- svm(V2~., data = trainset, kernel="radial", gamma=0.001,
          cost=10)
print(model)
summary(model)
```

Then, its output is the following:

```
Call:
   svm(formula = V2∽., data = trainset, kernel = "radial",
           gamma = 0.001, cost = 10)

Parameters:
    SVM-Type:  C-classification
    SVM-Kernel:  radial
    cost:  10
    gamma:  0.001

Number of Support Vectors:  82    ( 41 41 )

Number of Classes:  2

Levels:
    B M
```

Predict Method for SVM.

In order to predict using the classifier obtained before, the command we must use is called *predict.svm*. Command "predict" is applied on an object of class "svm", as can be seen in the following example.

If we write "`?predict.svm`" in the R console, we obtain

```
## S3 method for class 'svm':
predict((object, newdata, decision.values = FALSE,
probability = FALSE, ..., na.action = na.omit))
```

This function predicts values based on a model trained by "svm". The output of command "predict.svm" is a vector of predicted values (for classification a vector of labels, for density estimation a logical vector). If *decision.value* is TRUE, the vector gets a "decision.value" attribute containing an $n \times c$ matrix of all c classifiers decision values. There are $\frac{k(k-1)}{2}$ classifiers, where k is the number of classes. The colnames of the matrix indicate the labels of the two classes. If *probability* is TRUE, the vector gets *probabilities* attribute containing an $n \times k$ matrix of the class probabilities.

Some tools are used on the command:

- `object`: describe an object of class "*svm*", created by command "*svm*"
- `newdata`: is an object containing the new input data: either a matrix or a sparse matrix. A vector will be transformed to a $n \times 1$ matrix
- `decision.values`: is a logical controlling whether the decision values of al binary classifiers computed in multiclass classification shall be computed and returned
- `probability`: is a logical indicating whether class probabilities should be computed and returned. only possible if the model was fitted with the probability option enabled
- `na.action`: is a function to specify the action to be taken if NA's are found. The default action is "*na.omit*", which leads to rejection of cases with missing values on any require variable

Using the test set as before, we will implement an example of "predict.svm" command, in order to predict the classes for each sample:

```
prediction <- predict(model, testset[,-1])
```

The -1 is because the label column to intance classes, V2, is in the first column. To produce the confusion matrix type:

```
tab <- table(pred = prediction, true = testset[,1])
```

The confusion matrix obtained with those predictions is:

		True	
		M	B
Prediction	M	55	0
	B	5	110

Here, there were 110 benign instances in the test set and all of them were predicted as benign ones. On the other hand, there were 60 malign instances in the test set, 55 of which were predicted correctly, and 5 as benign instances, and thus erroneously.

Let us consider the following values:

- **TP**: true positive, i.e., malign instances correctly predicted
- **TN**: true negative, i.e., benign instances correctly predicted

- **FP**: false positive, i.e., benign instances predicted as malign
- **FN**: false negative, i.e., malign instances predicted as benign
- $|\mathbf{P}|$: total of malign instances
- $|\mathbf{N}|$: total of benign instances

Then, some of the performance values introduced in Chapter 2, can be calculated as follows

- Sensitivity= $\dfrac{TP}{|P|}$
- Specificity= $\dfrac{TN}{|N|}$
- Accuracy= $\dfrac{TP+TN}{|P|+|N|}$
- Precision=$\dfrac{TP}{TP+FP}$

Hence, for this particular problem we have:

- Sensitivity= $\dfrac{55}{60} = 0.916$
- Specificity= $\dfrac{110}{110} = 1$
- Accuracy= $\dfrac{55+110}{60+110} = 0.971$
- Precision=$\dfrac{55}{55+0} = 1$

Plot SVM Objects.

In this last part, the so-called command *plot.svm* is explained. In order to make a plot of the SVM output, we simply apply the command "plot" on a class "svm".

As before, if we write `?plot.svm` in the R console, we could see:

```
## S3 method for class 'svm'
plot(x, data, formula, fill = TRUE, grid = 50, slice = list(),
symbolPalette = palette(), svSymbol = "x", dataSymbol = "o", ...)
```

As we can see above, the command "plot.svm" has the following arguments:

- `x`: object of class "svm"
- `data`: the data which we will visualize. Should be the same used for fitting
- `formula`: the formula selecting the visualized two dimensions
- `fill`: a switch indicating whether a contour plot for the class regions should be added
- `slice`: only needed if more than two variables are used. Is a list of named values for the dimension held constant
- `symbolPalette`: the color palette used for the class the data points and support vectors belong to
- `svSymbol`: symbols used for support vectors
- `dataSymbol`: symbols used for data points

We can use additional graphics parameters. "plot.svm" generates a scatter plot of the input data of a svm fit for classification models by highlighting the classes and support vectors. Optionally, it draws a filled contour plot of the class regions.

3.5 SVM in AMPL

The use of *AMPL* in needed, since new additional constraints will be added to SVM in order to guarantee a minimum value in a given performance measurement in an independent or validation dataset. This can be difficult to implement in R, and thus an optimization problem in integer numbers is defined and solved.

The following AMPL model was used to solve the following dual SVM problem

$$\begin{array}{ll} \max & \sum_{i\in\mathcal{I}} \lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}} \lambda_i\lambda_j y_i y_j K(x_i,x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}} y_i\lambda_i = 0 \\ & 0 \le \lambda_i \le C \ \forall i \in \mathcal{I} \end{array}$$

In particular, the cases when $K(x_i,x_j) = x_i^t x_j$ and $K(x_i,x_j) = exp\left(-\gamma\|x_i - x_j\|^2\right)$ are considered.

Model document

```
# Number of training samples
param l;

# Number of independent samples
param l2;

# Number of variables
param n;

# C parameter of SVM
param C;

# Parameters of RBF kernel
param gamma;

# Label vectors
param y {1..l};
param y2 {1..l2};

# Data (variables) matrices
param x{1..l,1..n};
param x2{1..l2,1..n};

# Dual problem variables a (lambda) and C constraint
var a{1..l} >= 0, <= C;

# Optimization problem (linear kernel)
maximize svmlin:  sum{i in 1..l}a[i]-0.5*
                  sum{i in 1..l,j in 1..l}a[i]*a[j]*y[i]*y[j]*
                  (sum{k in 1..n}x[i,k]*x[j,k]);
```

```
# Optimization problem (RBF kernel)
maximize svmrad:  sum {i in 1..l}a[i]-0.5*
                  sum{i in 1..l,j in 1..l}a[i]*a[j]*y[i]*y[j]*
                  exp(-gamma*(sum{k in i..n}(x[i,k]-x[j,k])^2));

# Constraints
s.t.  rest1:  sum{i in 1..l}a[i]*y[i]=0;
```

Command document

```
# We solve SVM optimization problem
solve;

# And display the a variables
display a;

# Initialization of some required parameters
param b{1..n} default 0;
param b0 default 0;
param counter default 0;
param primea{1..l} default 0;
param confmat{1..2,1..2} default 0;
param prediction{1..l2} default 0;

# b (beta) vector from linear kernel
let {j in 1..n} b[j]:= sum{i in 1..l} a[i]*y[i]*x[i,j];

# Make a counter of support vectors and display results
for{i in 1..l} {
      if a[i] >=10^(-8) then let counter:= counter+1;
      if a[i] >=10^(-8) then let primea[i]:=1;
}
display counter;
display primea;

# b0 from linear kernel
let b0 := 1/counter*(sum{ i in 1..l} primea[i]*
      (y[i]- sum{j in 1..l} a[j]*y[j]*
      (sum{k in 1..n} x[j,k]*x[i,k])));

# b0 from radial kernel
```

```
# let b0 := 1/counter*(sum{ i in 1..l} primea[i]*
      (y[i]- sum{j in 1..l} a[j]*y[j]*
      (sum{k in 1..n}exp(-gamma*(sum{k in i..n}(x[i,k]-x[j,k])^2)))));

# Show b (beta) from linear kernel
display b;

# Show b0 from linear/radial kernel
display b0;

# Prediction linear kernel
let {j in 1..l2} prediction[j]:=b0+sum{i in 1..n}b[i]*x2[j,i];

# prediction radial kernel
# let {jj in 1..l2} prediction[jj]:=
      # b0+sum{j in 1..l}primea[j]*a[j]*y[j]*
      # exp(-gamma*(sum{k in i..n}(x2[jj,k]-x[j,k])^2))

# Prediction as -1 or +1
let {j in 1..l2} prediction[j]:=
      if prediction[j] <= 0 then -1 else 1;

# Show the prediction vector
display prediction;

# Construction of confusion matrix
for {i in 1..l2}{
      if prediction[i] == y2[i] and prediction[i]=1
      then let confmat[1,1]:= confmat[1,1] + 1;
}
for {i in 1..l2}{
      if prediction[i] == y2[i] and prediction[i]=-1
      then let confmat[2,2]:= confmat[2,2] + 1;
}
for {i in 1..l2}{
      if prediction[i]*y2[i]==-1 and prediction[i]=1
      then let confmat[1,2]:= confmat[1,2] + 1;
}
for {i in 1..l2}{
      if prediction[i]*y2[i]==-1 and prediction[i]=-1
      then let confmat[2,1]:= confmat[2,1] + 1;
```

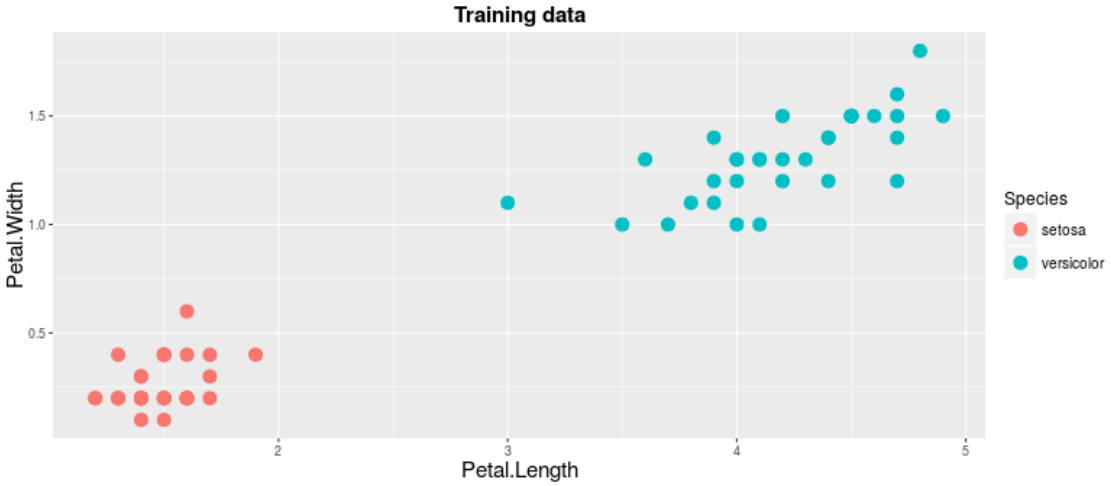
```
}

# Show confusion matrix
printf "confmat is the confusion matrix of our classification
      problem.  confmat[1,1] is TP, confmat[2,2] is TN, confmat[1,2]
      is FP and confmat[2,1] is FN \n";
display confmat;
```

3.5.1 Examples. Iris Data

Now, we will make three different experiments with the code above. As data, we will use *iris data* from R. As such data is composed of three classes and we focus in binary classification, we will select two of them for each example. In the first example we choose *setosa* and *versicolor*, and in the second and third ones, *versicolor* and *virginica*. In order to have an easier representation of the data, we only use two variables, both to represent the data and to build the SVM classifier. The first example, is a linearly separable case. The second one, is an almost linearly separable case. In the last one, data are completely "mixed".

In the three examples, we will select (randomly) as training set a 66% of the total amount of samples, and the rest will be used as test set.

Linearly separable data. Hard margin SVM. The next picture shows the data we have select as training.



Once the optimization problem is solved, the values of b and b_0 (β and β_0, respectively) from the primal problem are obtained:

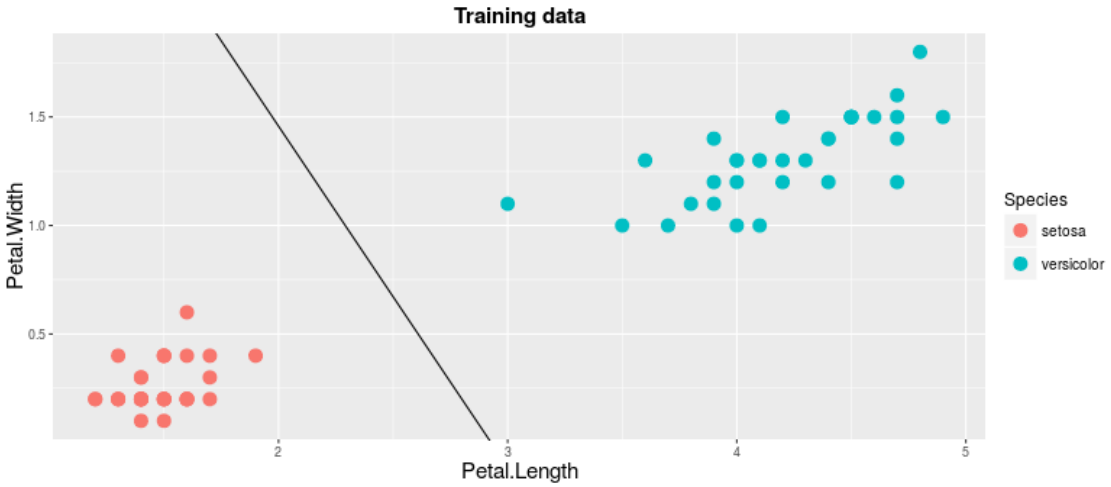
```
b = (-1.29412,-0.823529)^t
b0 = 3.78824
```

And thus, the separating line results

$$X_2 = \frac{-1.29412}{0.823529} X_1 + \frac{3.78824}{0.823529}$$

where $X_1 \equiv$ Petal.Length and $X_2 \equiv$ Petal.Width.

In the picture below we can observe how such line separates perfectly the data.



In fact, the confusion table obtained when our previous optimization problem is solved is

		True	
		setosa	versicolor
Prediction	setosa	33	0
	versicolor	0	33

Hence

$$\text{CCR}_{\text{setosa}} = \text{CCR}_{\text{versicolor}} = \text{Accuracy} = \text{Precision} = 1,$$

where CRC_k is the *Correct Clasification Rate* for class k, introduced in Chapter 2.

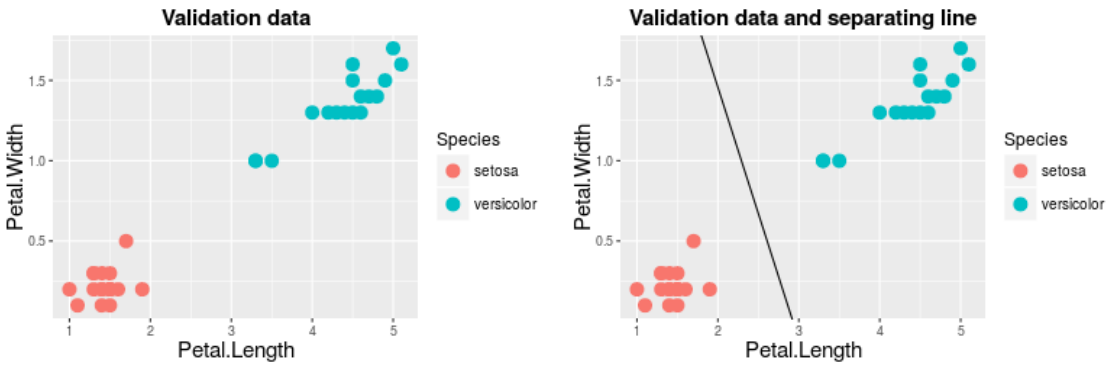
Also, after making the validation in the test data, we obtain the confusion table

		True	
		setosa	versicolor
Prediction	setosa	17	0
	versicolor	0	17

so the performance measurements in this validation dataset, as before in the training one, are

$$\text{CCR}_{\text{setosa}} = \text{CCR}_{\text{versicolor}} = \text{Accuracy} = \text{Precision} = 1.$$

In fact, the results we obtain in the validation dataset can be seen in the following picture, which confirms the values above.



Nonlinearly separable data. Soft margin SVM. Let us present a second example but in non-separable data this time. The C value chosen in this experiment will be $C = 1$.

The picture below shows the new dataset selected as training.

Once the optimization problem is solved again, the values of b and b_0 (β and β_0) from the primal problem are:

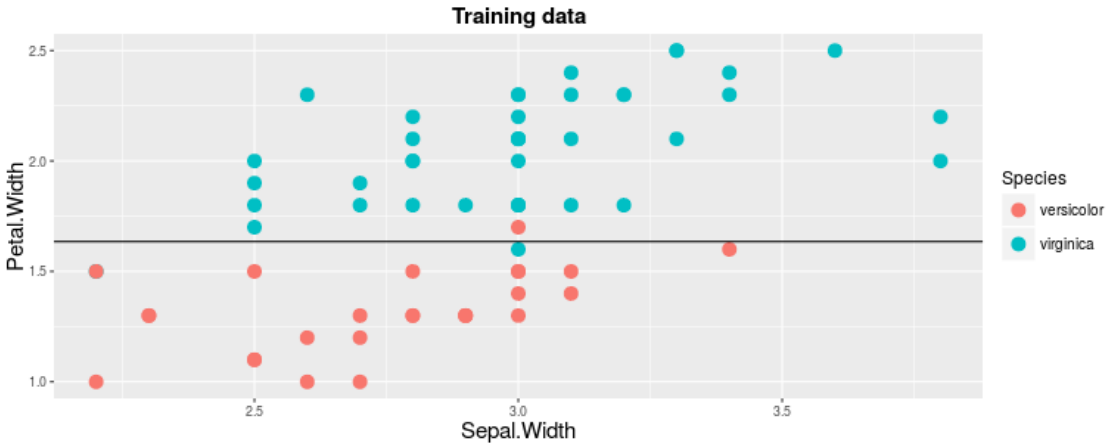
```
b = (-1.91555e-09,6.66667)^t
b0 = -10.9
```

And thus, the separating line results

$$X_2 = \frac{-1.91555e-09}{-6.66667} X_1 + \frac{-10.9}{-6.66667}$$

where $X_1 \equiv$ Sepal.Width and $X_2 \equiv$ Petal.Width.

Now, we can observe in the figure below that data is not separated at all.



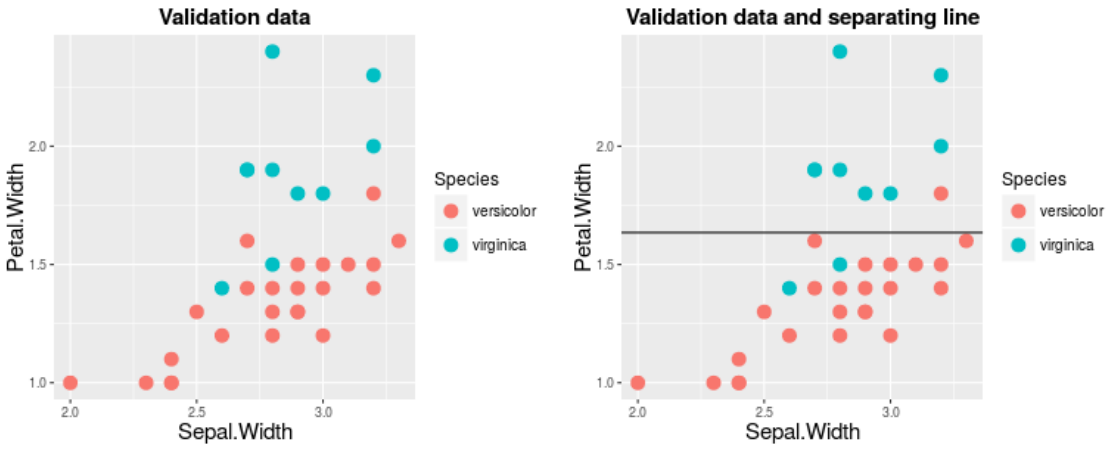
In fact, the confusion table obtained for this training data is

		True	
		versicolor	virginica
Prediction	versicolor	25	2
	virginica	1	38

So

- $\text{CCR}_{\text{versicolor}} = \dfrac{25}{26} = 0,9615$
- $\text{CCR}_{\text{virginica}} = \dfrac{38}{40} = 0,95$
- $\text{Precision} = \dfrac{25}{27} = 0,9259$
- $\text{Accuracy} = \dfrac{63}{66} = 0,9545.$

Now, as before, we validate the model in the test data. What we got can be seen in the picture below



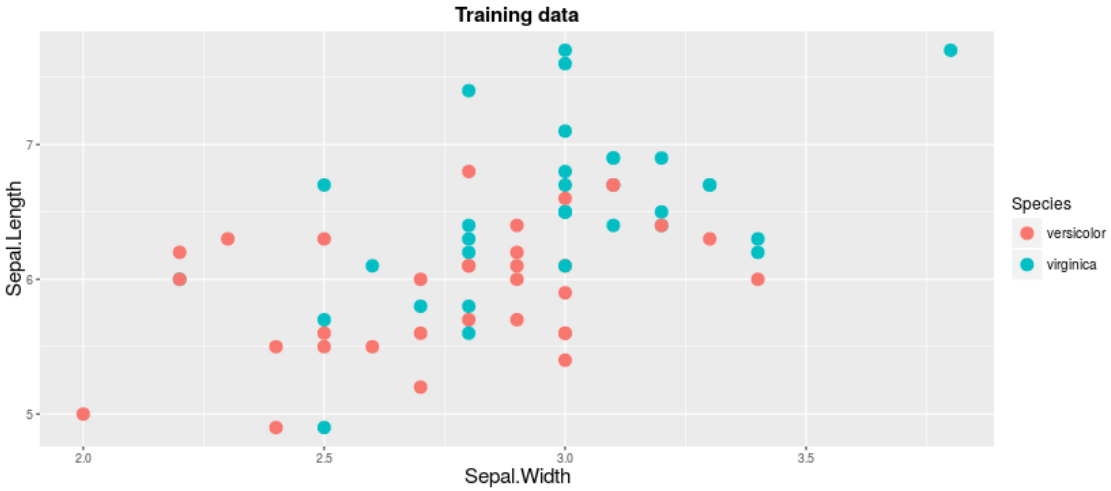
In this case, the confusion table is

		True	
		versicolor	virginica
Prediction	versicolor	23	2
	virginica	1	8

and thus

- $\text{CCR}_{\text{versicolor}} = \dfrac{23}{24} = 0,9583$
- $\text{CCR}_{\text{virginica}} = \dfrac{8}{10} = 0,8$
- $\text{Precision} = \dfrac{23}{25} = 0,92$
- $\text{Accuracy} = \dfrac{31}{34} = 0,9118.$

Nonlinearly separable data. Radial basis function SVM. The next picture shows the data we have select as training.



As we can observe, data are completely mixed. Because of this fact, we use the radial basis function in this case.

As before, we run the AMPL code (with $C = 1$ and $\gamma = 0.5$), and the confusion table obtained is

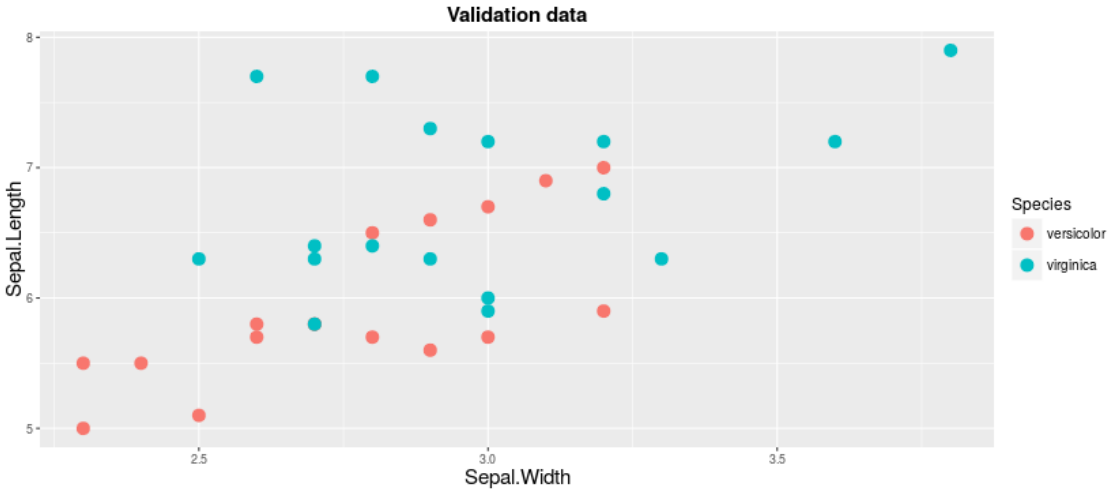
		True	
		versicolor	virginica
Prediction	versicolor	26	10
	virginica	7	23

and thus

- $\text{CCR}_{\text{versicolor}} = \dfrac{26}{33} = 0,7879$

- $\text{CCR}_{\text{virginica}} = \dfrac{23}{33} = 0,697$
- $\text{Precision} = \dfrac{26}{36} = 0,7222$
- $\text{Accuracy} = \dfrac{49}{66} = 0,7424.$

Once again, we validate this model with the test dataset. Such dataset is represented in the next picture.



As in the training dataset case, data is completely mixed. Now, the confusion table is

		True	
		versicolor	virginica
Prediction	versicolor	12	6
	virginica	5	11

and thus

- $\text{CCR}_{\text{versicolor}} = \dfrac{12}{17} = 0,7059$
- $\text{CCR}_{\text{virginica}} = \dfrac{11}{17} = 0,6471$
- $\text{Precision} = \dfrac{12}{18} = 0,6667$
- $\text{Accuracy} = \dfrac{23}{34} = 0,6765.$

3.5.2 Examples. Breast Cancer Wisconsin data

Now, as in Sections 3.4, we will use Breast Cancer Wisconsin data, [9]. In this section, however, we will use only two of the 30 total variables that there are in the complete dataset. The reason of why we do that is because we want a "bad" separation between the data in order to compare the results obtained here with the ones that we will obtain with the new formulation of SVM presented in Chapter 4, in which we impose to have a minimum value of a given perfomance index.

As before, data will be separated into the training set (70%) and the test set (30%). Also, as the data we will use is only composed by two variables, they can be represented in the plane, giving

First, we apply a linear SVM to this data. For that, we select the best C in a range of C of the form $\{2^{-6}, 2^{-5}, \ldots, 2^5, 2^6\}$.

For each C the different perfomance values obtained are

C	TP	TN	FP	FN	$Sens$	$Spec$	Acc	$Prec$
2^{-6}	102	0	68	0	1	0	0.6	0.6
2^{-5}	102	0	68	0	1	0	0.6	0.6
2^{-4}	102	3	65	0	1	0.04	0.62	0.61
2^{-3}	93	30	38	9	0.91	0.44	0.72	0.71
2^{-2}	97	24	44	5	0.95	0.35	0.71	0.69
2^{-1}	101	5	63	1	0.99	0.07	0.62	0.62
2^{0}	97	22	46	5	0.95	0.32	0.7	0.68
2^{1}	94	27	41	8	0.92	0.4	0.71	0.7
2^{2}	84	41	27	18	0.82	0.6	0.74	0.76
2^{3}	84	41	27	18	0.82	0.6	0.74	0.76
2^{4}	84	42	26	18	0.82	0.62	0.74	0.76
2^{5}	84	42	26	18	0.82	0.62	0.74	0.76
2^{6}	84	42	26	18	0.82	0.62	0.74	0.76

Now, we apply a radial SVM. As in the linear case, we will use a set of ranges, this time for both C and γ, the parameters of radial SVM, so actually we will use a grid. The grid will be composed by the values of $C = \{2^{-5}, 2^{-4}, \ldots, 2^{4}, 2^{5}\}$ and $\gamma = \{2^{-5}, 2^{-4}, \ldots, 2^{4}, 2^{5}\}$.

Now, the results obtained are shown in the following table

C	γ	**TP**	**TN**	**FP**	**FN**	**Sens**	**Spec**	**Acc**	**Prec**
2^{-5}	2^{-5}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-5}	2^{-4}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-5}	2^{-3}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-5}	2^{-2}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-5}	2^{-1}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-5}	2^{0}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-5}	2^{1}	96	12	56	6	0.94	0.18	0.64	0.63
2^{-5}	2^{2}	90	17	51	12	0.88	0.25	0.63	0.64
2^{-5}	2^{3}	84	24	44	18	0.82	0.35	0.64	0.66
2^{-5}	2^{4}	82	32	36	20	0.80	0.47	0.67	0.69
2^{-5}	2^{5}	87	34	34	15	0.85	0.50	0.71	0.72
2^{-4}	2^{-5}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-4}	2^{-4}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-4}	2^{-3}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-4}	2^{-2}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-4}	2^{-1}	102	3	65	0	1.00	0.04	0.62	0.61
2^{-4}	2^{0}	91	14	54	11	0.89	0.21	0.62	0.63

Continued on next page

Table 3.2 – *Continued from previous page*

C	γ	**TP**	**TN**	**FP**	**FN**	**Sens**	**Spec**	**Acc**	**Prec**
2^{-4}	2^{1}	82	20	48	20	0.80	0.29	0.60	0.63
2^{-4}	2^{2}	77	30	38	25	0.75	0.44	0.63	0.67
2^{-4}	2^{3}	77	38	30	25	0.75	0.56	0.68	0.72
2^{-4}	2^{4}	79	39	29	23	0.77	0.57	0.69	0.73
2^{-4}	2^{5}	79	39	29	23	0.77	0.57	0.69	0.73
2^{-3}	2^{-5}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-3}	2^{-4}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-3}	2^{-3}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-3}	2^{-2}	99	4	64	3	0.97	0.06	0.61	0.61
2^{-3}	2^{-1}	90	14	54	12	0.88	0.21	0.61	0.63
2^{-3}	2^{0}	81	20	48	21	0.79	0.29	0.59	0.63
2^{-3}	2^{1}	75	31	37	27	0.74	0.46	0.62	0.67
2^{-3}	2^{2}	74	43	25	28	0.73	0.63	0.69	0.75
2^{-3}	2^{3}	73	43	25	29	0.72	0.63	0.68	0.74
2^{-3}	2^{4}	73	40	28	29	0.72	0.59	0.66	0.72
2^{-3}	2^{5}	74	41	27	28	0.73	0.60	0.68	0.73
2^{-2}	2^{-5}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-2}	2^{-4}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-2}	2^{-3}	98	6	62	4	0.96	0.09	0.61	0.61
2^{-2}	2^{-2}	90	15	53	12	0.88	0.22	0.62	0.63
2^{-2}	2^{-1}	79	24	44	23	0.77	0.35	0.61	0.64
2^{-2}	2^{0}	74	36	32	28	0.73	0.53	0.65	0.70
2^{-2}	2^{1}	74	43	25	28	0.73	0.63	0.69	0.75
2^{-2}	2^{2}	70	43	25	32	0.69	0.63	0.66	0.74
2^{-2}	2^{3}	69	44	24	33	0.68	0.65	0.66	0.74
2^{-2}	2^{4}	71	42	26	31	0.70	0.62	0.66	0.73
2^{-2}	2^{5}	72	43	25	30	0.71	0.63	0.68	0.74
2^{-1}	2^{-5}	102	0	68	0	1.00	0.00	0.60	0.60
2^{-1}	2^{-4}	98	6	62	4	0.96	0.09	0.61	0.61
2^{-1}	2^{-3}	90	15	53	12	0.88	0.22	0.62	0.63
2^{-1}	2^{-2}	77	30	38	25	0.75	0.44	0.63	0.67
2^{-1}	2^{-1}	74	43	25	28	0.73	0.63	0.69	0.75
2^{-1}	2^{0}	71	43	25	31	0.70	0.63	0.67	0.74
2^{-1}	2^{1}	70	44	24	32	0.69	0.65	0.67	0.74
2^{-1}	2^{2}	68	44	24	34	0.67	0.65	0.66	0.74
2^{-1}	2^{3}	67	44	24	35	0.66	0.65	0.65	0.74
2^{-1}	2^{4}	70	43	25	32	0.69	0.63	0.66	0.74
2^{-1}	2^{5}	70	44	24	32	0.69	0.65	0.67	0.74

Continued on next page

Table 3.2 – *Continued from previous page*

C	γ	**TP**	**TN**	**FP**	**FN**	**Sens**	**Spec**	**Acc**	**Prec**
2^0	2^{-5}	98	6	62	4	0.96	0.09	0.61	0.61
2^0	2^{-4}	90	15	53	12	0.88	0.22	0.62	0.63
2^0	2^{-3}	75	33	35	27	0.74	0.49	0.64	0.68
2^0	2^{-2}	70	44	24	32	0.69	0.65	0.67	0.74
2^0	2^{-1}	67	45	23	35	0.66	0.66	0.66	0.74
2^0	2^0	67	45	23	35	0.66	0.66	0.66	0.74
2^0	2^1	67	45	23	35	0.66	0.66	0.66	0.74
2^0	2^2	67	45	23	35	0.66	0.66	0.66	0.74
2^0	2^3	67	45	23	35	0.66	0.66	0.66	0.74
2^0	2^4	70	44	24	32	0.69	0.65	0.67	0.74
2^0	2^5	70	44	24	32	0.69	0.65	0.67	0.74
2^1	2^{-5}	90	15	53	12	0.88	0.22	0.62	0.63
2^1	2^{-4}	74	38	30	28	0.73	0.56	0.66	0.71
2^1	2^{-3}	67	46	22	35	0.66	0.68	0.66	0.75
2^1	2^{-2}	58	47	21	44	0.57	0.69	0.62	0.73
2^1	2^{-1}	63	47	21	39	0.62	0.69	0.65	0.75
2^1	2^0	66	46	22	36	0.65	0.68	0.66	0.75
2^1	2^1	67	45	23	35	0.66	0.66	0.66	0.74
2^1	2^2	67	45	23	35	0.66	0.66	0.66	0.74
2^1	2^3	67	46	22	35	0.66	0.68	0.66	0.75
2^1	2^4	70	44	24	32	0.69	0.65	0.67	0.74
2^1	2^5	70	46	22	46	0.60	0.68	0.63	0.76
2^2	2^{-5}	74	42	26	28	0.73	0.62	0.68	0.74
2^2	2^{-4}	64	47	21	38	0.63	0.69	0.65	0.75
2^2	2^{-3}	57	50	18	45	0.56	0.74	0.63	0.76
2^2	2^{-2}	55	50	18	47	0.54	0.74	0.62	0.75
2^2	2^{-1}	58	49	19	44	0.57	0.72	0.63	0.75
2^2	2^0	65	46	22	37	0.64	0.68	0.65	0.75
2^2	2^1	67	45	23	35	0.66	0.66	0.66	0.74
2^2	2^2	67	45	23	35	0.66	0.66	0.66	0.74
2^2	2^3	67	46	22	35	0.66	0.68	0.66	0.75
2^2	2^4	70	44	24	32	0.69	0.65	0.67	0.74
2^2	2^5	69	46	22	33	0.68	0.68	0.68	0.76
2^3	2^{-5}	60	48	20	42	0.59	0.71	0.64	0.75
2^3	2^{-4}	54	50	10	48	0.53	0.83	0.64	0.84
2^3	2^{-3}	53	51	17	49	0.52	0.75	0.61	0.76
2^3	2^{-2}	54	50	18	48	0.53	0.74	0.61	0.75
2^3	2^{-1}	58	49	19	44	0.57	0.72	0.63	0.75

Continued on next page

Table 3.2 – *Continued from previous page*

C	γ	**TP**	**TN**	**FP**	**FN**	**Sens**	**Spec**	**Acc**	**Prec**
2^3	2^0	65	47	21	37	0.64	0.69	0.66	0.76
2^3	2^1	67	46	22	35	0.66	0.68	0.66	0.75
2^3	2^2	67	45	23	35	0.66	0.66	0.66	0.74
2^3	2^3	67	46	22	35	0.66	0.68	0.66	0.75
2^3	2^4	70	44	24	32	0.69	0.65	0.67	0.74
2^3	2^5	69	46	22	33	0.68	0.68	0.68	0.76
2^4	2^{-5}	0	68	0	102	0.00	1.00	0.40	-
2^4	2^{-4}	0	68	0	102	0.00	1.00	0.40	-
2^4	2^{-3}	0	68	0	102	0.00	1.00	0.40	-
2^4	2^{-2}	1	68	0	101	0.01	1.00	0.41	1.00
2^4	2^{-1}	16	68	0	86	0.16	1.00	0.49	1.00
2^4	2^0	41	60	8	61	0.40	0.88	0.59	0.84
2^4	2^1	53	50	18	49	0.52	0.74	0.61	0.75
2^4	2^2	58	49	19	44	0.57	0.72	0.63	0.75
2^4	2^3	63	47	21	39	0.62	0.69	0.65	0.75
2^4	2^4	67	47	21	35	0.66	0.69	0.67	0.76
2^4	2^5	67	46	22	35	0.66	0.68	0.66	0.75
2^5	2^{-5}	0	68	0	102	0.00	1.00	0.40	-
2^5	2^{-4}	0	68	0	102	0.00	1.00	0.40	-
2^5	2^{-3}	0	68	0	102	0.00	1.00	0.40	-
2^5	2^{-2}	1	68	0	101	0.01	1.00	0.41	1.00
2^5	2^{-1}	16	68	0	86	0.16	1.00	0.49	1.00
2^5	2^0	41	60	8	61	0.40	0.88	0.59	0.84
2^5	2^1	53	50	18	49	0.52	0.74	0.61	0.75
2^5	2^2	58	49	19	44	0.57	0.72	0.63	0.75
2^5	2^3	63	47	21	39	0.62	0.69	0.65	0.75
2^5	2^4	67	47	21	35	0.66	0.69	0.67	0.76
2^5	2^5	67	46	22	35	0.66	0.68	0.66	0.75

Chapter 4

Adding new constraints in SVM

In this chapter we make an attempt to improve the classification of the SVM (in one or more of the performance indices such as sensitivity, specificity,...). In order to do that, a new set of constraints is proposed. Before we make the proposal of new constraints, some theory is presented.

4.1 Theoretical motivation

Let us suppose that we have available a non-empty set of data Ω,

$$\Omega = \{u_i = (x_i, y_i) : i = 1, \ldots, n\}$$

where each $u_i \in \Omega$ has two components: with $x_i \in \mathbb{R}^r$ an r-dimensional vector of predictor variables and $y_i \in \{-1, +1\}$ two given classes of u_i

$$(x_i, y_i)_{i=1}^n \qquad \text{where } y_i \in \{-1, 1\} \text{ and } x \in \mathbb{R}^r.$$

Let p be

- the probability of correct classification of any sample in a class, i.e.,

$$p = Sensitivity = \frac{\text{true positive}}{\text{true positive + false negative}}$$

$$p = Specificity = \frac{\text{true negative}}{\text{true negative + false positive}}$$

- a predictive value

$$p = PPV = \frac{\text{true positive}}{\text{true positive + false positive}}$$

$$p = NPV = \frac{\text{true negative}}{\text{true negative + false negative}};$$

- or another measurement, such as e.g. the accuracy, defined as the overall probability of correct classification

Our objective is to get a classifier such that $p \geq p_0$, where p_0 is a fixed desired value. As the distribution of the data is unknown, we cannot evaluate p. Instead, given a independent sample $\mathcal{I}nd$, we can calculate an estimator $\hat{p}(\mathcal{I}nd)$ of p, and try to impose $\hat{p}(\mathcal{I}nd) \geq p_0$.

But if we have a classifier which gets $\hat{p}(\mathcal{I}nd) \geq p_0$ for a given dataset, it does not mean it will happen in other different samples: it can not be guaranteed that the probability of correct classification to be greater than a certain value. Instead, for a random variable X we have several ways to obtain an inequality of the form

$$P(\overline{X} - E[X] \geq c) \leq g(c),$$

which we can be solved for c and make $g(c) = \alpha$. Then, we have

$$P(E[X] \geq \overline{X} - c_\alpha) \geq 1 - \alpha,$$

i.e., an $1 - \alpha\%$ confidence interval (CI) for $E[X]$ is $(\overline{X} - c_\alpha, 1)$.

Although the theory showed above about the CI seems not to have a strong relationship with our desire of getting a classifier such that $p \geq p_0$, it has: if we have a set of independent variables $\{Y_i\}_{i \in S}$, $Y_i \sim Be(p)$ $\forall i$ such that

$$Y_i = \begin{cases} 1, & \text{if record } i \text{ is well classified} \\ 0, & \text{otherwise} \end{cases}$$

then

$$\overline{Y} = \frac{1}{|S|} \sum_{i \in S} Y_i = \hat{p}(\mathcal{I}nd),$$

and

$$E[Y] = p = E[\hat{p}(\mathcal{I}nd)],$$

where S is a set of samples in which we are focusing, such as e.g. the samples that correspond to the positive class (if we want a CI for the sensitivity), and where $\overline{Y}$ and $E[Y]$ act, respectively, like the previous $\overline{X}$ and $E[X]$.

So what we can do is to search for a classifier such that, taking an independent dataset, we can ensure that the value p_0 is in the CI for $E[Y] = p$. If we want that the lower endpoint of such CI is greater than p_0, then $\overline{Y} - c_\alpha = \hat{p}(\mathcal{I}nd) - c_\alpha \geq p_0$, thus $\hat{p}(\mathcal{I}nd) \geq p_0 + c_\alpha = p_0'$.

In the literature, there exist many ways to obtain the mentioned CI. Confidence intervals for the mean of a random variable, Y, are based on the mean of i.i.d. samples of that random variable:

$$\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i, \qquad Y_1, Y_2, \dots \ i.i.d.$$

Now, in the next subsections, we present tree ways to obtain such CI.

4.1.1 Markov's inequality

Markov's inequality may be used to construct a fixed-width CI for $p = E[\hat{p}(\mathcal{I}nd)]$. This inequality makes relatively mild assumptions on the distribution of the random variable.

Let X be a nonnegative random variable and $\overline{X} + c \geq 0$, then

$$EX = \int x dF(x) = \int_0^{E[X]+c} x dF(x) + \int_{E[X]+c}^{\infty} x dF(x) \geq 0 + (E[X]+c)P(X > E[X]+c) \Rightarrow$$

$$\Rightarrow P(X > E[X] + c) \leq \frac{E[X]}{E[X] + c}$$

In our case, we have $X = \overline{Y} = \hat{p}(\mathcal{I}nd)$, and $E[X] = E[\overline{Y}] = E[\hat{p}(\mathcal{I}nd)] = p$, so

$$P(\hat{p}(\mathcal{I}nd) \geq p + c) \leq \frac{p}{p + c},$$

thus

$$P(p \geq \hat{p}(\mathcal{I}nd) - c) \geq 1 - \frac{p}{p + c}.$$

Then, if we want an $1 - \alpha\%$ CI for p, it should be

$$\alpha = \frac{p}{p + c} \Rightarrow (p + c)\alpha = p \Rightarrow p\alpha + c\alpha = p \Rightarrow c\alpha = p(1 - \alpha) \Rightarrow c = \frac{p(1 - \alpha)}{\alpha}$$

and the CI resulting is

$$(\hat{p}(\mathcal{I}nd) - \frac{p(1-\alpha)}{\alpha}, 1).$$

So, if we want the lower endpoint of the CI greater than p_0, it must be $p_0 \leq \hat{p}(\mathcal{I}nd) - \frac{p(1-\alpha)}{\alpha}$, hence $\hat{p}(\mathcal{I}nd) \geq p_0 + \frac{p(1-\alpha)}{\alpha} = p_0'$.

4.1.2 Central Limit Theorem (CLT)

Another option is to use the *CLT*, which describes how the distribution of $\hat{p}(\mathcal{I}nd) = \overline{Y}$ approaches a Gaussian distribution as $n \to \infty$, i.e.

$$\hat{p} \approx \mathcal{N}\left(p, \frac{p(1-p)}{n}\right),$$

where $p = E[\hat{p}(\mathcal{I}nd)]$.

Then,

$$P(\hat{p}(\mathcal{I}nd) - p > c) \approx P(\mathcal{N}\left(p, \frac{p(1-p)}{n}\right) - p > c) =$$

$$= P(\mathcal{N}\left(0, \frac{p(1-p)}{n}\right) > c) = P(\mathcal{N}(0,1) > \frac{c\sqrt{n}}{\sqrt{p(1-p)}}) \leq$$

$$\leq P(\mathcal{N}(0,1) > 2c\sqrt{n})$$

Which implies that

$$P(p \geq \hat{p}(\mathcal{I}nd) - c) \geq P(\mathcal{N}(0,1) \leq 2c\sqrt{n})$$

Therefore if we want to obtain an $1-\alpha\%$ CI for p, it must be

$$P(\mathcal{N}(0,1) \leq 2c\sqrt{n}) = 1 - \alpha \Rightarrow 2c\sqrt{n} = \Phi^{-1}(1-\alpha) \Rightarrow$$

$$\Rightarrow c = \frac{\Phi^{-1}(1-\alpha)}{2\sqrt{n}}$$

and the CI that we obtain is

$$(\hat{p}(\mathcal{I}nd) - \frac{\Phi^{-1}(1-\alpha)}{2\sqrt{n}}, 1)$$

And if we impose that the lower endpoint of the previous CI is greater than p_0, then $p_0 \leq \hat{p}(\mathcal{I}nd) - \frac{\Phi^{-1}(1-\alpha)}{2\sqrt{n}} \Rightarrow \hat{p}(\mathcal{I}nd) \geq p_0 + \frac{\Phi^{-1}(1-\alpha)}{2\sqrt{n}} = p_0'$.

4.1.3 Hoeffding's inequality

The last method we are going to discuss is based on Hoeffding's inequality, [8]. Here, we suppose $X_1, \ldots, X_n$ i.i.d., with $0 \leq X_i \leq 1 \ \forall i$. Hence, our case is in the conditions of this inequality, because we consider the random variables $\{Y_i\}_{i \in S}$ mentioned before. Then, for any $c > 0$, Hoeffding's inequality says

$$P(\overline{X} - E[X] \geq c) \leq e^{-2nc^2}$$

In the case that concerns us, we have $\overline{X} = \overline{Y} = \hat{p}(\mathcal{I}nd)$ and $E[X] = E[Y] = p$, so

$$P(\hat{p}(\mathcal{I}nd) - p \geq c) \leq e^{-2nc^2}$$

so

$$P(p \geq \hat{p}(\mathcal{I}nd) - c) \geq 1 - e^{-2nc^2}$$

And if we want to get an $1 - \alpha\%$ CI for p as before, then

$$1 - e^{-2nc^2} = 1 - \alpha \Rightarrow e^{-2nc^2} = \alpha \Rightarrow -2nc^2 = \log \alpha \Rightarrow c^2 = \frac{\log \alpha}{-2n} \underset{c>0}{\Longrightarrow} c = \sqrt{\frac{\log \alpha}{-2n}}$$

and the CI we get is

$$(\hat{p}(\mathcal{I}nd) - \sqrt{\frac{\log \alpha}{-2n}}, 1)$$

Therefore, if we impose as before, that the lower endpoint of this CI is greater than p_0, then $p_0 \leq \hat{p}(\mathcal{I}nd) - \sqrt{\frac{\log \alpha}{-2n}} \Rightarrow \hat{p}(\mathcal{I}nd) \geq p_0 + \sqrt{\frac{\log \alpha}{-2n}} = p_0'$.

4.2 General framework

Suppose that we have available a non-empty set of training data Ω_1, where each $u_i \in \Omega_1$ has two components:

$$\Omega_1 = \{u_i = (x_i, y_i) : i = 1, \ldots, n\}$$

with $x_i \in \mathbb{R}^r$ an r-dimensional vector of predictor variables and $y_i \in \{-1, +1\}$ two given classes of u_i

$$(x_i, y_i)_{i=1}^n \qquad \text{where } y_i \in \{-1, 1\} \text{ and } x \in \mathbb{R}^r.$$

In addition, let us consider another set of data, Ω_2, independent from the previous one, formed by points of the form

$$\Omega_2 = \{u_i' = (x_i, y_i) : i = n + 1, \ldots, n'\}$$

with $x_i \in \mathbb{R}^r$ an r-dimensional vector of predictor variables and $y_i \in \{-1, +1\}$ two given classes of u'_i ($\forall i = n+1, \ldots, n'$), as before.

Furthermore, let $\mathcal{I}$ and $\mathcal{I}nd$ be, respectively, the indices set of the data in Ω_1 and Ω_2.

As we have studied in the previous chapter, SVM can be expressed as the following optimization problem in general:

$$\begin{array}{ll} \max & \sum_{i\in\mathcal{I}} \lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}} \lambda_i\lambda_j y_i y_j K(x_i, x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}} y_i\lambda_i = 0 \\ & 0 \leq \lambda_i \leq C \ \forall i \in \mathcal{I} \end{array}$$

Since SVM is not more than an optmization problem, new constraints can be added. So, as an example, we can consider some performance measurements in the independent set, and impose their values to be greater than a certain threshold. Some examples of such performance measurements can be

- A given sensitivity or specificity
- Probability of correct and incorrect classification
- Sum of probabilities (e.g., sensitivity + specificity)
- $\cdots$

So we want a new formulation of SVM, simply adding some new constraints, to get a SVM of the form

$$\begin{array}{ll} \min & \sum_{i\in\mathcal{I}} \lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}} \lambda_i\lambda_j y_i y_j K(x_i, x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}} y_i\lambda_i = 0 \\ & 0 \leq \lambda_i \leq C \ \forall i \in \mathcal{I} \\ & \left\{ \begin{array}{l} \mu_1(u'_i, \lambda) \geq a_1 \\ \vdots \\ \mu_l(u'_i, \lambda) \geq a_l \end{array} \right. \end{array}$$

where $(\mu_i)_{i=1}^l$ are some performance measures and $(a_i)_{i=1}^l$ are their fixed lower bounds.

It is very important to remark that, in the linear kernel case, it is not completely necessary the dual formulation of SVM if we want to impose the value of any perfomance measurement to be greater than a given value. This fact can be observed in the later subsections.

4.2.1 m samples of independent set well classified

In order to classify in the correct class some of the instances in the independent dataset Ω_2, a new constraint must be imposed. First, we introduce some new values z_i, $i \in \mathcal{I}nd$, defined as:

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

We consider first the values z_i fixed.

For simplicity, we consider the linearly nonseparable case as basis. So the original primal optimization problem is given by

$$\begin{array}{ll} \min_{\beta_0,\beta,\varepsilon_i} & \frac{1}{2}\beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1,\ \forall i \in \mathcal{I} \\ & \beta_0 \in \mathbb{R},\ \beta \in \mathbb{R}^r \\ & \varepsilon_i \geq 0, \forall i \in \mathcal{I} \end{array}$$

In this way, the new primal formulation of the desired SMV results

$$\begin{array}{ll} \min_{\beta_0,\beta,\varepsilon_i} & \frac{1}{2}\beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i + M\sum_{i\in\mathcal{I}nd}\varepsilon_i z_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1,\ \forall i \in \mathcal{I} \\ & \beta_0 \in \mathbb{R},\ \beta \in \mathbb{R}^r \\ & \varepsilon_i \geq 0, \forall i \in \mathcal{I} \\ & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1,\ \forall i \in \mathcal{I}nd \\ & \varepsilon_i \geq 0, \forall i \in \mathcal{I}nd \end{array}$$

where M is sufficiently large.

We can group some of the constraints, and thus obtaining

$$\begin{array}{ll} \min_{\beta_0,\beta,\varepsilon_i} & \frac{1}{2}\beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i + M\sum_{i\in\mathcal{I}nd}\varepsilon_i z_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1,\ \forall i \in \mathcal{I}\cup\mathcal{I}nd \\ & \beta_0 \in \mathbb{R},\ \beta \in \mathbb{R}^r \\ & \varepsilon_i \geq 0, \forall i \in \mathcal{I}\cup\mathcal{I}nd. \end{array}$$

We now build its dual problem. In order to do this, we need to use the KKT method. Fixed $\lambda = (\lambda_1, \ldots, \lambda_n, \lambda_{n+1}, \ldots, \lambda_{n'})^t \geq 0$ and $\eta = (\eta_1, \ldots, \eta_n, \eta_{n+1}, \ldots, \eta_{n'})^t \geq 0$, let $\mathcal{L}(\beta_0, \beta, \lambda, \eta, \varepsilon_i)$ be the Lagrange function, defined as follows:

$$\mathcal{L}(\beta_0, \beta, \lambda, \eta, \varepsilon_i) =$$

$$= \frac{1}{2}\beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i + M\sum_{i\in\mathcal{I}nd}\varepsilon_i z_i - \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i[y_i(\beta_0+\beta^t x_i)-(1-\varepsilon_i)] - \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\eta_i\varepsilon_i$$

and proceeding as the KKT method says,

$$\frac{\partial}{\partial\beta}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\varepsilon_i): \quad \beta - \textstyle\sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i y_i x_i = 0$$

$$\frac{\partial}{\partial\beta_0}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\varepsilon_i): \quad -\textstyle\sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i y_i = 0$$

$$\frac{\partial}{\partial\varepsilon_i}\mathcal{L}(\beta,\beta_0,\lambda,\eta,\varepsilon_i): \quad \begin{cases} C-\lambda_i-\eta_i=0, & \forall i\in\mathcal{I} \\ Mz_i-\lambda_i-\eta_i=0, & \forall i\in\mathcal{I}nd \end{cases}$$

Substituting in the equation $\mathcal{L}(\beta_0,\beta,\lambda,\eta,\varepsilon_i)$, we obtain the dual of the problem above formulated. Hence:

$$\begin{array}{ll} \max_{\lambda,\eta} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i\lambda_j y_i y_j x_i^t x_j \\ \text{subject to:} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd} y_i\lambda_i = 0 \\ & \lambda_i = \begin{cases} C-\eta_i, & \forall i\in\mathcal{I} \\ Mz_i-\eta_i, & \forall i\in\mathcal{I}nd \end{cases} \\ & \eta_i,\lambda_i\geq 0,\ \forall i\in\mathcal{I}\cup\mathcal{I}nd \end{array}$$

Equivalently, the problem above can be formulated as

$$\begin{array}{ll} \max_{\lambda} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i\lambda_j y_i y_j x_i^t x_j \\ \text{subject to:} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd} y_i\lambda_i = 0 \\ & \lambda_i \leq C, \forall i\in\mathcal{I} \\ & \lambda_i \leq Mz_i, \forall i\in\mathcal{I}nd \\ & \lambda_i \geq 0, \forall i\in\mathcal{I}nd\cup\mathcal{I} \end{array}$$

But this is the formulation when the z_i are previously fixed. We can reformulate the previous optimization problem considering z_i, $i\in\mathcal{I}nd$ as binary variables, so we do not know which points are going to use the hard or soft margins and impose that at least m points of the independent set are going to be well classified, i.e, m points (at least) will use hard margin. This leads us to the following problem:

$$\begin{array}{ll} \max_{\lambda,z} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i\lambda_j y_i y_j x_i^t x_j \\ \text{subject to:} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd} y_i\lambda_i = 0 \\ & \lambda_i \leq C, \forall i\in\mathcal{I} \\ & \lambda_i \leq Mz_i, \forall i\in\mathcal{I}nd \\ & \sum_{i\in\mathcal{I}nd} z_i \geq m \\ & \lambda_i \geq 0, \forall i\in\mathcal{I}nd\cup\mathcal{I} \\ & z_i\in\{0,1\}, \forall i\in\mathcal{I}nd. \end{array}$$

Furthermore we can generalize the previous formulation. Using the kernel trick as in *Section* 3.2.1, we get the following optimization problem

$$\begin{array}{ll} \max_{\lambda,z} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i\lambda_j y_i y_j K(x_i,x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd} y_i\lambda_i = 0 \\ & \lambda_i \leq C, \forall i \in \mathcal{I} \\ & \lambda_i \leq M z_i, \forall i \in \mathcal{I}nd \\ & \sum_{i\in\mathcal{I}nd} z_i \geq m \\ & \lambda_i \geq 0, \forall i \in \mathcal{I}\cup\mathcal{I}nd \\ & z_i \in \{0,1\}, \forall i \in \mathcal{I}nd \end{array}$$

This problem is, as the standard SVM, concave quadratic with linear constraints. However, it has, together with the continuous variables λ_i, the binary variables z_i, which makes the problem much harder to solve.

As mentioned at the begining of this section, in the linear case we can use the primal formulation of SVM. It results:

$$\begin{array}{ll} \min & \beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1,\ \forall i \in \mathcal{I} \\ & y_j(\beta_0 + \beta^t x_j) \geq -M(1-z_j),\ \forall j \in \mathcal{I}nd \\ & \sum_{i\in\mathcal{I}nd} z_i \geq m \\ & \varepsilon_i \geq 0,\ \forall i \in \mathcal{I} \\ & z_j \in \{0,1\},\ \forall j \in \mathcal{I}nd, \end{array}$$

where z_i and M are defined as before.

4.2.2 Specifying a minimum sensitivity and specificity

From the previous formulation, we can include other constraints in order to get a desired value of sensitivity and specificity.

Having z_i defined as before, it is easy to check that the number of samples correctly classify in the -1 and $+1$ are, respectively, $\sum_{i\in\mathcal{I}nd}\frac{z_i(1-y_i)}{2}$ and $\sum_{i\in\mathcal{I}nd}\frac{z_i(y_i+1)}{2}$. As the total number of positive and negative samples are, respectively, $\sum_{i\in\mathcal{I}nd}\frac{y_i+1}{2}$ and $\sum_{i\in\mathcal{I}nd}\frac{1-y_i}{2}$, we obtain:

Formulation to specify a minimum sensitivity

$$
\begin{array}{ll}
\max_{\lambda,z} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i\lambda_j y_i y_j K(x_i,x_j) \\
\text{subject to:} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd} y_i\lambda_i = 0 \\
& \lambda_i \le C, \forall i \in \mathcal{I} \\
& \lambda_i \le M z_i, \forall i \in \mathcal{I}nd \\
& \dfrac{\sum_{i\in\mathcal{I}nd}\frac{z_i(y_i+1)}{2}}{\sum_{i\in\mathcal{I}nd}\frac{y_i+1}{2}} \ge a_1 \\
& \lambda_i \ge 0, \forall i \in \mathcal{I}\cup\mathcal{I}nd \\
& z_i \in \{0,1\}, \forall i \in \mathcal{I}nd.
\end{array}
$$

In the linear kernel case it can be also formulated as

$$
\begin{array}{ll}
\min & \beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i \\
\text{subject to:} & y_i(\beta_0+\beta^t x_i)+\varepsilon_i \ge 1,\ \forall i\in\mathcal{I} \\
& y_j(\beta_0+\beta^t x_j) \ge -M(1-z_j),\ \forall j\in\mathcal{I}nd \\
& \dfrac{\sum_{i\in\mathcal{I}nd}\frac{z_i(y_i+1)}{2}}{\sum_{i\in\mathcal{I}nd}\frac{y_i+1}{2}} \ge a_1 \\
& \varepsilon_i \ge 0,\ \forall i\in\mathcal{I} \\
& z_j \in \{0,1\},\ \forall j\in\mathcal{I}nd.
\end{array}
$$

Formulation to specify a minimum specificity

$$
\begin{array}{ll}
\max_{\lambda,z} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i\lambda_j y_i y_j K(x_i,x_j) \\
\text{subject to:} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd} y_i\lambda_i = 0 \\
& \lambda_i \le C, \forall i \in \mathcal{I} \\
& \lambda_i \le M z_i, \forall i \in \mathcal{I}nd \\
& \dfrac{\sum_{i\in\mathcal{I}nd}\frac{z_i(1-y_i)}{2}}{\sum_{i\in\mathcal{I}nd}\frac{1-y_i}{2}} \ge a_2 \\
& \lambda_i \ge 0, \forall i \in \mathcal{I}\cup\mathcal{I}nd \\
& z_i \in \{0,1\}, \forall i \in \mathcal{I}nd.
\end{array}
$$

Again, in the linear kernel case, it can be written as

$$\begin{array}{ll} \min & \beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0+\beta^t x_i)+\varepsilon_i \geq 1,\ \forall i\in\mathcal{I} \\ & y_j(\beta_0+\beta^t x_j) \geq -M(1-z_j),\ \forall j\in\mathcal{I}nd \\ & \dfrac{\sum_{i\in\mathcal{I}nd}\dfrac{z_i(y_i+1)}{2}}{\sum_{i\in\mathcal{I}nd}\dfrac{y_i+1}{2}} \geq a_2 \\ & \varepsilon_i \geq 0,\ \forall i\in\mathcal{I} \\ & z_j\in\{0,1\},\ \forall j\in\mathcal{I}nd. \end{array}$$

4.2.3 Specifying a minimum accuracy

As the total number of samples well classified is $\sum_{i\in Ind} z_i$ and the total number of samples in the independent sample is n', the acuracy is given by the quotient of those two values. Thus, the formulation to specify a minimum value of accuracy is given by

$$\begin{array}{ll} \max_{\lambda,z} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i - \dfrac{1}{2}\sum_{i,j\in\mathcal{I}\cup\mathcal{I}nd}\lambda_i\lambda_j y_i y_j K(x_i,x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}\cup\mathcal{I}nd} y_i\lambda_i = 0 \\ & \lambda_i \leq C, \forall i\in\mathcal{I} \\ & \lambda_i \leq M z_i, \forall i\in\mathcal{I}nd \\ & \dfrac{\sum_{i\in Ind} z_i}{n'} \geq a_3 \\ & \lambda_i \geq 0, \forall i\in\mathcal{I}\cup\mathcal{I}nd \\ & z_i\in\{0,1\}, \forall i\in\mathcal{I}nd. \end{array}$$

Once again, in the linear case this is equivalent to

$$\begin{array}{ll} \min & \beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0+\beta^t x_i)+\varepsilon_i \geq 1,\ \forall i\in\mathcal{I} \\ & y_j(\beta_0+\beta^t x_j) \geq -M(1-z_j),\ \forall j\in\mathcal{I}nd \\ & \dfrac{\sum_{i\in Ind} z_i}{n'} \geq a_3 \\ & \varepsilon_i \geq 0,\ \forall i\in\mathcal{I} \\ & z_j\in\{0,1\},\ \forall j\in\mathcal{I}nd. \end{array}$$

4.3 AMPL

A common formulation in AMPL of the linear kernel SVM above can be expressed as

```
# Number of training samples
param l;

   # Number of test/independent samples
param l2;

# Number of variables
param n;

# C parameter of SVM
param C;

# Large M
param M;

# Parameter b SVM
param b;

# Labels vectors
param y {1..l};
param y2 {1..l2};

# Data (variables) matrices
param x{1..l,1..n};
param x2{1..l2,1..n};

# Other SVM variables
var w{1..n};
var eps{1..l} >= 0;

# Variables if hard margin for test samples
var z{1..l2} binary;

# Optimization problem (linear kernel)
minimize svm:  sum{i in 1..n}w[i]*w[i] + C*sum{j in 1..l}(eps[j]);

# Constraints
subject to r1{i in 1..l}:  y[i]*((sum{j in 1..n}w[j]*x[i,j])+b)+xi[i]>=1;
```

- Constraint of Amount

```
 subject to r2{i in 1..l2}:
                  y2[i]*(sum{j in 1..n}w[j]*x2[i,j]+b)>= -(1-z[i])*M;
subject to restAm:  (sum{i in 1..l2} z[i])>= a1
```

- Constraint of Accuracy

```
 subject to r2{i in 1..l2}:
                  y2[i]*(sum{j in 1..n}w[j]*x2[i,j]+b)>= -(1-z[i])*M;
subject to restAcc:  (sum{i in 1..l2} z[i])/l2 >= a2
```

- Constraint of Sensitivity

```
 subject to r3{i in 1..l2 :  y[i]=1}:
                  y2[i]*(sum{j in 1..n}w[j]*x2[i,j]+b)>= -(1-z[i])*M;
s.t.  restSens:  (sum{i in 1..l2 :  y[i]=1}(z[i]*(y2[i]+1)/2))/
                  (sum{i in 1..l2}((y2[i]+1)/2)) >= a3;
```

- Constraint of Specificity

```
 subject to r4{i in 1..l2 :  y[i]=-1}:
                  y2[i]*(sum{j in 1..n}w[j]*x2[i,j]+b)>= -(1-z[i])*M;
s.t.  restSpec:  (sum{i in 1..l2 :  y[i]=-1}(z[i]*(1-y2[i])/2))/
                  (sum{i in 1..l2}((1-y2[i])/2))>= a4;
```

Note that the run script for this model is the same as the one we presented in Section 3.5.

4.3.1 Breast Cancer Wisconsin data

Now, as we did in Sections 3.4 and 3.5.2, we will use Breast Cancer Wisconsin data, from [9].

In order to make the experiment comparable to the previous one made in Section 3.5.2, we will use only the two variables selected in such section. As we mentioned, the reason of why we did that was because we wanted a "bad" separation between the data (and thus, a "bad" classification) in order to compare the results obtained here, with the new formulation of SVM (in which we impose to have, e.g., a minimum value of accuracy), with the results we got with the classical SVM.

Imposing a minimum value of Specificity

As we must predict if a given patient has a benign or a malign cancer, let us imagine that we want to have a minimum number of patients with bening cancer well classified (remember that in Section 3.5.2 such number

tended to be low). So in this case we want to impose a minimum value for the specificity in an independent sample, in particular we take $Specificity \geq 0.5$.

Due to this fact, we use the formulation in Subsection 4.2.2 and thus the AMPL constraints for Specificity exclusively.

Now we can apply to this data a linear SVM. For that, we use range of C of the form $\{2^{-6}, 2^{-5}, \ldots, 2^5, 2^6\}$. Also, in order to make this linear SVM more flexible, we include "b" (β_0) as a parameter to tune; hence using a range of the form $\{-0.6, -0.2, \ldots, 2.2, 2.6\}$.

For each pair of C and b the different perfomance values obtained are given in the following table. When both TP and FP are zero, then the precision (Prec) cannot be calculated and thus a symbol "-" is used.

C	**b**	**TP**	**TN**	**FP**	**FN**	**Sens**	**Spec**	**Acc**	**Prec**
2^{-6}	-0.6	0	68	0	102	0.00	1.00	0.40	-
2^{-6}	-0.2	11	55	13	91	0.11	0.81	0.39	0.46
2^{-6}	0.2	79	34	34	23	0.77	0.50	0.66	0.70
2^{-6}	0.6	79	34	34	23	0.77	0.50	0.66	0.70
2^{-6}	1	79	34	34	23	0.77	0.50	0.66	0.70
2^{-6}	1.4	79	34	34	23	0.77	0.50	0.66	0.70
2^{-6}	1.8	79	34	34	23	0.77	0.50	0.66	0.70
2^{-6}	2.2	79	34	34	23	0.77	0.50	0.66	0.70
2^{-6}	2.6	79	34	34	23	0.77	0.50	0.66	0.70
2^{-5}	-0.6	0	66	2	102	0.00	0.97	0.39	0.00
2^{-5}	-0.2	31	35	33	71	0.30	0.51	0.39	0.48
2^{-5}	0.2	79	34	34	23	0.77	0.50	0.66	0.70
2^{-5}	0.6	79	34	34	23	0.77	0.50	0.66	0.70
2^{-5}	1	79	34	34	23	0.77	0.50	0.66	0.70
2^{-5}	1.4	79	34	34	23	0.77	0.50	0.66	0.70
2^{-5}	1.8	79	34	34	23	0.77	0.50	0.66	0.70
2^{-5}	2.2	79	34	34	23	0.77	0.50	0.66	0.70
2^{-5}	2.6	79	34	34	23	0.77	0.50	0.66	0.70
2^{-4}	-0.6	12	53	15	90	0.12	0.78	0.38	0.44
2^{-4}	-0.2	33	34	34	69	0.32	0.50	0.39	0.49
2^{-4}	0.2	91	34	34	11	0.89	0.50	0.74	0.73
2^{-4}	0.6	81	34	34	21	0.79	0.50	0.68	0.70
2^{-4}	1	79	34	34	23	0.77	0.50	0.66	0.70
2^{-4}	1.4	79	34	34	23	0.77	0.50	0.66	0.70

Continued on next page

Table 4.1 – *Continued from previous page*

C	b	TP	TN	FP	FN	Sens	Spec	Acc	Prec
2^{-4}	1.8	79	34	34	23	0.77	0.50	0.66	0.70
2^{-4}	2.2	79	34	34	23	0.77	0.50	0.66	0.70
2^{-4}	2.6	79	34	34	23	0.77	0.50	0.66	0.70
2^{-3}	-0.6	29	34	34	73	0.28	0.50	0.37	0.46
2^{-3}	-0.2	33	34	34	69	0.32	0.50	0.39	0.49
2^{-3}	0.2	91	34	34	11	0.89	0.50	0.74	0.73
2^{-3}	0.6	85	34	34	17	0.83	0.50	0.70	0.71
2^{-3}	1	81	34	34	21	0.79	0.50	0.68	0.70
2^{-3}	1.4	81	34	34	21	0.79	0.50	0.68	0.70
2^{-3}	1.8	79	34	34	23	0.77	0.50	0.66	0.70
2^{-3}	2.2	79	34	34	23	0.77	0.50	0.66	0.70
2^{-3}	2.6	79	34	34	23	0.77	0.50	0.66	0.70
2^{-2}	-0.6	31	34	34	71	0.30	0.50	0.38	0.48
2^{-2}	-0.2	33	34	34	69	0.32	0.50	0.39	0.49
2^{-2}	0.2	83	34	34	19	0.81	0.50	0.69	0.71
2^{-2}	0.6	91	34	34	11	0.89	0.50	0.74	0.73
2^{-2}	1	88	34	34	14	0.86	0.50	0.72	0.72
2^{-2}	1.4	85	34	34	17	0.83	0.50	0.70	0.71
2^{-2}	1.8	85	34	34	17	0.83	0.50	0.70	0.71
2^{-2}	2.2	85	34	34	17	0.83	0.50	0.70	0.71
2^{-2}	2.6	81	34	34	21	0.79	0.50	0.68	0.70
2^{-1}	-0.6	31	34	34	71	0.30	0.50	0.38	0.48
2^{-1}	-0.2	35	34	67	34	0.51	0.34	0.41	0.34
2^{-1}	0.2	82	34	34	20	0.80	0.50	0.68	0.71
2^{-1}	0.6	93	34	34	9	0.91	0.50	0.75	0.73
2^{-1}	1	91	34	34	11	0.89	0.50	0.74	0.73
2^{-1}	1.4	91	34	34	11	0.89	0.50	0.74	0.73
2^{-1}	1.8	90	34	34	12	0.88	0.50	0.73	0.73
2^{-1}	2.2	88	34	34	14	0.86	0.50	0.72	0.72
2^{-1}	2.6	88	34	34	14	0.86	0.50	0.72	0.72
2^{0}	-0.6	33	34	34	69	0.32	0.50	0.39	0.49
2^{0}	-0.2	37	34	34	65	0.36	0.50	0.42	0.52
2^{0}	0.2	73	34	34	29	0.72	0.50	0.63	0.68
2^{0}	0.6	86	34	34	16	0.84	0.50	0.71	0.72
2^{0}	1	93	34	34	9	0.91	0.50	0.75	0.73
2^{0}	1.4	91	34	34	11	0.89	0.50	0.74	0.73
2^{0}	1.8	91	34	34	11	0.89	0.50	0.74	0.73
2^{0}	2.2	91	34	34	11	0.89	0.50	0.74	0.73

Continued on next page

Table 4.1 – *Continued from previous page*

C	b	TP	TN	FP	FN	Sens	Spec	Acc	Prec
2^0	2.6	91	34	34	11	0.89	0.50	0.74	0.73
2^1	-0.6	33	34	34	69	0.32	0.50	0.39	0.49
2^1	-0.2	38	34	34	64	0.37	0.50	0.42	0.53
2^1	0.2	69	34	34	33	0.68	0.50	0.61	0.67
2^1	0.6	86	34	34	16	0.84	0.50	0.71	0.72
2^1	1	86	34	34	16	0.84	0.50	0.71	0.72
2^1	1.4	93	34	34	9	0.91	0.50	0.75	0.73
2^1	1.8	93	34	34	9	0.91	0.50	0.75	0.73
2^1	2.2	93	34	34	9	0.91	0.50	0.75	0.73
2^1	2.6	91	34	34	11	0.89	0.50	0.74	0.73
2^2	-0.6	33	34	34	69	0.32	0.50	0.39	0.49
2^2	-0.2	38	34	34	64	0.37	0.50	0.42	0.53
2^2	0.2	68	34	34	34	0.67	0.50	0.60	0.67
2^2	0.6	83	34	34	19	0.81	0.50	0.69	0.71
2^2	1	86	34	34	16	0.84	0.50	0.71	0.72
2^2	1.4	93	34	34	9	0.91	0.50	0.75	0.73
2^2	1.8	93	34	34	9	0.91	0.50	0.75	0.73
2^2	2.2	93	34	34	9	0.91	0.50	0.75	0.73
2^2	2.6	93	34	34	9	0.91	0.50	0.75	0.73
2^3	-0.6	33	34	34	69	0.32	0.50	0.39	0.49
2^3	-0.2	38	34	34	64	0.37	0.50	0.42	0.53
2^3	0.2	68	34	34	34	0.67	0.50	0.60	0.67
2^3	0.6	82	34	34	20	0.80	0.50	0.68	0.71
2^3	1	86	34	34	16	0.84	0.50	0.71	0.72
2^3	1.4	89	34	34	13	0.87	0.50	0.72	0.72
2^3	1.8	93	34	34	9	0.91	0.50	0.75	0.73
2^3	2.2	93	34	34	9	0.91	0.50	0.75	0.73
2^3	2.6	93	34	34	9	0.91	0.50	0.75	0.73
2^4	-0.6	33	34	34	69	0.32	0.50	0.39	0.49
2^4	-0.2	43	34	34	59	0.42	0.50	0.45	0.56
2^4	0.2	82	34	34	20	0.80	0.50	0.68	0.71
2^4	0.6	86	34	34	16	0.84	0.50	0.71	0.72
2^4	1	86	34	34	16	0.84	0.50	0.71	0.72
2^4	1.4	93	34	34	9	0.91	0.50	0.75	0.73
2^4	1.8	93	34	34	9	0.91	0.50	0.75	0.73
2^4	2.2	93	34	34	9	0.91	0.50	0.75	0.73
2^4	2.6	93	34	34	9	0.91	0.50	0.75	0.73
2^5	-0.6	33	34	34	69	0.32	0.50	0.39	0.49

Continued on next page

Table 4.1 – *Continued from previous page*

C	b	TP	TN	FP	FN	Sens	Spec	Acc	Prec
2^5	-0.2	41	34	34	61	0.40	0.50	0.44	0.55
2^5	0.2	67	34	34	35	0.66	0.50	0.59	0.66
2^5	0.6	82	34	34	20	0.80	0.50	0.68	0.71
2^5	1	86	34	34	16	0.84	0.50	0.71	0.72
2^5	1.4	86	34	34	16	0.84	0.50	0.71	0.72
2^5	1.8	93	34	34	9	0.91	0.50	0.75	0.73
2^5	2.2	93	34	34	9	0.91	0.50	0.75	0.73
2^5	2.6	93	34	34	9	0.91	0.50	0.75	0.73
2^6	-0.6	33	34	34	69	0.32	0.50	0.39	0.49
2^6	-0.2	44	34	34	58	0.43	0.50	0.46	0.56
2^6	0.2	67	34	34	35	0.66	0.50	0.59	0.66
2^6	0.6	82	34	34	20	0.80	0.50	0.68	0.71
2^6	1	86	34	34	16	0.84	0.50	0.71	0.72
2^6	1.4	86	34	34	16	0.84	0.50	0.71	0.72
2^6	1.8	93	34	34	9	0.91	0.50	0.75	0.73
2^6	2.2	93	34	34	9	0.91	0.50	0.75	0.73
2^6	2.6	93	34	34	9	0.91	0.50	0.75	0.73

Imposing a minimum value of Accruracy As before, we must predict if a given patient has a benign or a malign cancer. If we want to have a minimum number of patients well classified (without take into account the class), we must impose a minimum value for the accuracy in an independent sample, in particular we take $Accuracy \geq 0.75$.

Due to this fact, we use the formulation in Subsection 4.2.3 and thus the AMPL constraints for Accuracy.

Now we can apply to this data a linear SVM. For that, we use a range of C of the form $\{2^{-6}, 2^{-5}, \ldots, 2^5, 2^6\}$. Also, in order to make this linear SVM more flexible (as before), we include "b" (β_0) as a parameter to tune; hence using a range of the form $\{-0.6, -0.2, \ldots, 2.2, 2.6\}$.

In what follows, for each pair of C and b, the different perfomance values obtained are presented. When the constrained problem is unfeasible, the values are replaced by the symbol "-".

C	b	TP	TN	FP	FN	Sens	Spec	Acc	Prec
2^{-6}	-0.6	-	-	-	-	-	-	-	-

Continued on next page

Table 4.2 – *Continued from previous page*

C	b	TP	TN	FP	FN	Sens	Spec	Acc	Prec
2^{-6}	-0.2	-	-	-	-	-	-	-	-
2^{-6}	0.2	90	39	29	12	0.88	0.57	0.76	0.76
2^{-6}	0.6	89	40	28	13	0.87	0.59	0.76	0.76
2^{-6}	1	89	39	29	13	0.87	0.57	0.75	0.75
2^{-6}	1.4	90	39	29	12	0.88	0.57	0.76	0.76
2^{-6}	1.8	89	40	28	13	0.87	0.59	0.76	0.76
2^{-6}	2.2	89	40	28	13	0.87	0.59	0.76	0.76
2^{-6}	2.6	90	40	28	12	0.88	0.59	0.76	0.76
2^{-5}	-0.6	-	-	-	-	-	-	-	-
2^{-5}	-0.2	-	-	-	-	-	-	-	-
2^{-5}	0.2	90	39	29	12	0.88	0.57	0.76	0.76
2^{-5}	0.6	89	40	28	13	0.87	0.59	0.76	0.76
2^{-5}	1	89	39	29	13	0.87	0.57	0.75	0.75
2^{-5}	1.4	90	39	29	12	0.88	0.57	0.76	0.76
2^{-5}	1.8	89	40	28	13	0.87	0.59	0.76	0.76
2^{-5}	2.2	89	40	28	13	0.87	0.59	0.76	0.76
2^{-5}	2.6	90	40	28	12	0.88	0.59	0.76	0.76
2^{-4}	-0.6	-	-	-	-	-	-	-	-
2^{-4}	-0.2	-	-	-	-	-	-	-	-
2^{-4}	0.2	90	39	29	12	0.88	0.57	0.76	0.76
2^{-4}	0.6	89	40	28	13	0.87	0.59	0.76	0.76
2^{-4}	1	89	39	29	13	0.87	0.57	0.75	0.75
2^{-4}	1.4	89	39	29	13	0.87	0.57	0.75	0.75
2^{-4}	1.8	89	40	28	13	0.87	0.59	0.76	0.76
2^{-4}	2.2	89	40	28	13	0.87	0.59	0.76	0.76
2^{-4}	2.6	90	40	28	12	0.88	0.59	0.76	0.76
2^{-3}	-0.6	-	-	-	-	-	-	-	-
2^{-3}	-0.2	-	-	-	-	-	-	-	-
2^{-3}	0.2	91	38	30	11	0.89	0.56	0.76	0.75
2^{-3}	0.6	89	40	28	13	0.87	0.59	0.76	0.76
2^{-3}	1	89	39	29	13	0.87	0.57	0.75	0.75
2^{-3}	1.4	90	39	29	12	0.88	0.57	0.76	0.76
2^{-3}	1.8	89	40	28	13	0.87	0.59	0.76	0.76
2^{-3}	2.2	89	40	28	13	0.87	0.59	0.76	0.76
2^{-3}	2.6	89	39	29	13	0.87	0.57	0.75	0.75
2^{-2}	-0.6	-	-	-	-	-	-	-	-
2^{-2}	-0.2	-	-	-	-	-	-	-	-
2^{-2}	0.2	91	38	30	11	0.89	0.56	0.76	0.75

Continued on next page

Table 4.2 – *Continued from previous page*

C	b	TP	TN	FP	FN	Sens	Spec	Acc	Prec
2^{-2}	0.6	90	39	29	12	0.88	0.57	0.76	0.76
2^{-2}	1	90	39	29	12	0.88	0.57	0.76	0.76
2^{-2}	1.4	89	40	28	13	0.87	0.59	0.76	0.76
2^{-2}	1.8	89	40	28	13	0.87	0.59	0.76	0.76
2^{-2}	2.2	89	39	29	13	0.87	0.57	0.75	0.75
2^{-2}	2.6	90	40	28	12	0.88	0.59	0.76	0.76
2^{-1}	-0.6	-	-	-	-	-	-	-	-
2^{-1}	-0.2	-	-	-	-	-	-	-	-
2^{-1}	0.2	92	37	31	10	0.90	0.54	0.76	0.75
2^{-1}	0.6	92	38	30	10	0.90	0.56	0.76	0.75
2^{-1}	1	90	39	29	12	0.88	0.57	0.76	0.76
2^{-1}	1.4	90	39	29	12	0.88	0.57	0.76	0.76
2^{-1}	1.8	89	40	28	13	0.87	0.59	0.76	0.76
2^{-1}	2.2	89	40	28	13	0.87	0.59	0.76	0.76
2^{-1}	2.6	90	40	28	12	0.88	0.59	0.76	0.76
2^{0}	-0.6	-	-	-	-	-	-	-	-
2^{0}	-0.2	-	-	-	-	-	-	-	-
2^{0}	0.2	87	40	28	15	0.85	0.59	0.75	0.76
2^{0}	0.6	91	38	30	11	0.89	0.56	0.76	0.75
2^{0}	1	91	37	31	11	0.89	0.54	0.75	0.75
2^{0}	1.4	92	36	32	10	0.90	0.53	0.75	0.74
2^{0}	1.8	89	40	28	13	0.87	0.59	0.76	0.76
2^{0}	2.2	89	40	28	13	0.87	0.59	0.76	0.76
2^{0}	2.6	90	40	28	12	0.88	0.59	0.76	0.76
2^{1}	-0.6	-	-	-	-	-	-	-	-
2^{1}	-0.2	-	-	-	-	-	-	-	-
2^{1}	0.2	92	37	31	10	0.90	0.54	0.76	0.75
2^{1}	0.6	91	38	30	11	0.89	0.56	0.76	0.75
2^{1}	1	91	37	31	11	0.89	0.54	0.75	0.75
2^{1}	1.4	92	37	31	10	0.90	0.54	0.76	0.75
2^{1}	1.8	91	38	30	11	0.89	0.56	0.76	0.75
2^{1}	2.2	92	38	30	10	0.90	0.56	0.76	0.75
2^{1}	2.6	92	38	30	10	0.90	0.56	0.76	0.75
2^{2}	-0.6	-	-	-	-	-	-	-	-
2^{2}	-0.2	-	-	-	-	-	-	-	-
2^{2}	0.2	91	37	31	11	0.89	0.54	0.75	0.75
2^{2}	0.6	91	38	30	11	0.89	0.56	0.76	0.75
2^{2}	1	91	37	31	11	0.89	0.54	0.75	0.75

Continued on next page

Table 4.2 – *Continued from previous page*

C	b	TP	TN	FP	FN	Sens	Spec	Acc	Prec
2^2	1.4	92	37	31	10	0.90	0.54	0.76	0.75
2^2	1.8	91	38	30	11	0.89	0.56	0.76	0.75
2^2	2.2	91	38	30	11	0.89	0.56	0.76	0.75
2^2	2.6	91	37	31	11	0.89	0.54	0.75	0.75
2^3	-0.6	-	-	-	-	-	-	-	-
2^3	-0.2	-	-	-	-	-	-	-	-
2^3	0.2	91	37	31	11	0.89	0.54	0.75	0.75
2^3	0.6	91	38	30	11	0.89	0.56	0.76	0.75
2^3	1	91	37	31	11	0.89	0.54	0.75	0.75
2^3	1.4	92	37	31	10	0.90	0.54	0.76	0.75
2^3	1.8	91	38	30	11	0.89	0.56	0.76	0.75
2^3	2.2	91	38	30	11	0.89	0.56	0.76	0.75
2^3	2.6	91	37	31	11	0.89	0.54	0.75	0.75
2^4	-0.6	-	-	-	-	-	-	-	-
2^4	-0.2	-	-	-	-	-	-	-	-
2^4	0.2	92	37	31	10	0.90	0.54	0.76	0.75
2^4	0.6	91	38	30	11	0.89	0.56	0.76	0.75
2^4	1	91	37	31	11	0.89	0.54	0.75	0.75
2^4	1.4	92	37	31	10	0.90	0.54	0.76	0.75
2^4	1.8	91	38	30	11	0.89	0.56	0.76	0.75
2^4	2.2	91	38	30	11	0.89	0.56	0.76	0.75
2^4	2.6	91	37	31	11	0.89	0.54	0.75	0.75
2^5	-0.6	-	-	-	-	-	-	-	-
2^5	-0.2	-	-	-	-	-	-	-	-
2^5	0.2	87	40	28	15	0.85	0.59	0.75	0.76
2^5	0.6	92	38	30	10	0.90	0.56	0.76	0.75
2^5	1	91	37	31	11	0.89	0.54	0.75	0.75
2^5	1.4	92	37	31	10	0.90	0.54	0.76	0.75
2^5	1.8	91	38	30	11	0.89	0.56	0.76	0.75
2^5	2.2	91	38	30	11	0.89	0.56	0.76	0.75
2^5	2.6	91	37	31	11	0.89	0.54	0.75	0.75
2^6	-0.6	-	-	-	-	-	-	-	-
2^6	-0.2	-	-	-	-	-	-	-	-
2^6	0.2	92	37	31	10	0.90	0.54	0.76	0.75
2^6	0.6	91	38	30	11	0.89	0.56	0.76	0.75
2^6	1	91	37	31	11	0.89	0.54	0.75	0.75
2^6	1.4	92	37	31	10	0.90	0.54	0.76	0.75
2^6	1.8	91	38	30	11	0.89	0.56	0.76	0.75

Continued on next page

Table 4.2 – *Continued from previous page*

C	**b**	**TP**	**TN**	**FP**	**FN**	**Sens**	**Spec**	**Acc**	**Prec**
2^6	2.2	91	38	30	11	0.89	0.56	0.76	0.75
2^6	2.6	91	37	31	11	0.89	0.54	0.75	0.75

4.4 Conclusions

In this work we have analyzed SVM in its basic form, as well as the extension obtained when constraints are introduced to control different performance measures. The models are expressed as mathematical optimization problems, which have been programmed in R and in the modeling language AMPL to be used with a quadratic solver such as CPLEX. As a summary, we observe that improvements in the performance measures can be obtained by adding, as done here, constraints to the optimization problem of maximization of the margin. The price to pay is that, instead of quadratic concave maximization problems with linear constraints, integer variables are now introduced, leading to harder optimization problems.

Bibliography

[1] Emilio Carrizosa and Dolores Romero Morales. Supervised classification and mathematical optimization. *Computers & Operations Research*, 40(1):150–165, 2013.

[2] Rich Caruana and Alexandru Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78. ACM, 2004.

[3] A Casado-Reinaldos. Mathematical optimization and feature selection, 2015. Universidad de Sevilla. https://idus.us.es/xmlui/handle/11441/40792.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *The Journal of Machine Learning Research*, 5:1391–1415, 2004.

[6] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[7] A Izenman. *Modern multivariate statistical techniques*, volume 1. Springer, 2008.

[8] Lan Jiang and Fred J Hickernell. Guaranteed monte carlo methods for bernoulli random variables. *arXiv preprint arXiv:1411.1151*, 2014.

[9] M. Lichman. UCI machine learning repository, 2013.

[10] F Plastria and E Carrizosa. Gauge distances and median hyperplanes. *Journal of Optimization Theory and Applications*, 110(1):173–182, 2001.

[11] John C Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. In *nips*, volume 12, pages 547–553, 1999.