TRABAJO FIN DE GRADO

MATHEMATICAL OPTIMIZATION AND FEATURE SELECTION

Presented by:
Alejandro Casado Reinaldos

Supervisors:
DR. RAFAEL BLANQUERO BRAVO, Universidad de Sevilla
DR. EMILIO CARRIZOSA PRIEGO, Universidad de Sevilla

UNIVERSIDAD DE SEVILLA

June 21, 2015

Contents

Chapter 1

Introduction

Supervised classification is a common task in big data. It seeks procedures for classifying objects in a set Ω into a set $\mathcal{C}$ of classes, [7]. Supervised classification has been successfully applied in many different fields. Examples are found in text categorization, such as document indexing, webpage classification and spam filtering; biology and medicine, such as classification of gene expression data, homology detection, protein–protein interaction prediction, abnormal brain activity classification and cancer diagnosis; machine vision; agriculture; or chemistry, to cite a few fields.

Mathematical optimization has played a crucial role in supervised classification. Techniques from very diverse fields within mathematical optimization have been shown to be useful. Support Vector Machine (SVM) is one of the main exponents as application of the mathematical optimization to supervised classification. SVM is a state of the art method for supervised learning. For the two-class case, SVM aims at separating both classes by means of a hyperplane which maximizes the margin, i.e., the width of the band separating the two sets. This geometrical optimization problem can be written as a convex quadratic optimization problem with linear constraints, in principle solvable by any nonlinear optimization procedure.

In some applications the amount of features is huge and training SVM using the entire feature set would be computationally very expensive, while its outcome would lack from insight. This is, for instance, the case in gene expression and text categorization. In this sense, we talk about the combinatorial problem of selecting a best-possible set of features, discarding the remaining ones. It is called the feature selection problem.

In this work we analyze SVMs and how relevant features can be identified. In Chapter 2, we describe the SVM, first in the case of a linear kernel and then for the more interesting case of nonlinear kernels. We show how SVM can be handled in the statistical programme R. Then, some issues related with feature selection are described in Chapter 3.

Chapter 2

Support Vector Machine

2.1 Linear Support Vector Machine

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

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

with $x_i \in \mathbb{R}^r$, vector of predictors variables, and $y_i \in \{1, -1\}$ two given classes of u_i. We now have a non-empty set $\mathcal{I}$, which will be called the learning set. The learning set is composed of $u_i = (x_i, y_i)$, where y_i is given, $\forall\ i$. The binary 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 $\beta \in \mathbb{R}^r$ y $\beta_0 \in \mathbb{R}$ in order to construct a function $f : \mathbb{R}^r \rightarrow \mathbb{R}$ such that:

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

This function is called separation function, [16]. It 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 in $\mathcal{I}$, i.e. $(y_i = 1)$, are assigned to class 1 and negative points in $\mathcal{I}$, i.e. $(y_i = -1)$, to class -1. Points x with $f(x) = 0$ must be classified according to a predeterminied rule. It is defined with the system

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

2.1.1 The Linearly Separable Case

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

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

where β is the weight vector with norm $\|\beta\|$, and β_0 is the bias. If this hyperplane can separate the learning set of data into the two given classes 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

$$\begin{gathered} H^+ = \beta_0 + \beta^t x_i > 0,\ if\ y_i = 1 \\ H^- = \beta_0 + \beta^t x_i < 0,\ if\ y_i = -1 \end{gathered}$$

For separable sets, we have an infinite number of such hyperplanes. Consider any separating hyperplane. Let d_- be the shortest distance from the separating hyperplane to the nearest negative data point, and let d_+ be the shortest distance from the separating hyperplane to the nearest positive data point. We say that the hyperplane is an optimal separating hyperplane if we maximize the distance between the hyperplane and the closest observation.

In order to find the best separating hyperplane, we use a norm $\|.\|$ 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 $\|.\|$ be the Euclidean distance, Then, given x, we have

$$d_- = d(x, \{y : \beta_0 + \beta^t y \le 0\}) = \max \frac{\{\beta_0 + \beta^t x, 0\}}{\|\beta\|} \tag{2.1}$$

$$d_+ = d(x, \{y : \beta_0 + \beta^t y \ge 0\}) = \max \frac{\{-(\beta_0 + \beta^t x), 0\}}{\|\beta\|} \tag{2.2}$$

Proof. Let x be a fixed point, we have the following problem which formulates the distance between point x from the separating hyperplane:

$$\begin{array}{ll} \min & \|x - y\| \\ \text{subject to:} & (\beta_0 + \beta^t y) = 0 \end{array} \tag{2.3}$$

Equivalent to:

$$\begin{array}{ll} \min & \|x - y\|^2 \\ \text{subject to:} & (\beta_0 + \beta^t y) = 0 \end{array} \tag{2.4}$$

With the Euclidean distance we are in conditions to use the Karush-Kuhn-Tucker (KKT) conditions. Then let $\mathcal{L}(y, \lambda)$ be the Lagrange function defined as follows:

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

Proceeding as the method KKT says:

$\frac{\partial}{\partial y}\mathcal{L}(y,\lambda)$: $\quad 2(y-x)-\lambda\beta=0$

$2\beta^t(y-x)-\lambda\beta^t\beta=0$, $(\beta^t y=-\beta_0)$

$-2\beta_0-2\beta^t x=\lambda\beta^t\beta$

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

Replacing λ in equation of $\frac{\partial}{\partial y}\mathcal{L}(y,\lambda)$ and applying norm $\|.\|$:

$2(y-x)=\lambda\beta$

$\|y-x\|=\frac{|\lambda|}{2}\|\beta\|$

$|\frac{-2\beta_0-2\beta^t x}{\beta^t\beta}|\frac{\|\beta\|}{2}=\frac{|\beta_0+\beta^t x|}{\|\beta\|^2}\|\beta\|=\frac{|\beta_0+\beta^t x|}{\|\beta\|}$

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:

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

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

For an arbitrary norm, we can use the following result, which extends property, [24]:

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

$$d_{\|.\|}(x,H(\beta,\beta_0))=\begin{cases}\frac{\{\beta_0-\langle\beta;x\rangle\}}{\|\beta\|^\circ}, & when\ \langle\beta;x\rangle\leq\beta_0,\\ \\ \frac{\{\langle\beta;x\rangle-\beta_0\}}{\|\beta\|^\circ}, & when\ \langle\beta;x\rangle>\beta_0.\end{cases}$$

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

$$\|\beta\|^\circ = \begin{array}{ll} \max & u^t\beta \\ \text{subject to:} & \|u\| = 1. \end{array} \tag{2.5}$$

We have in the previous theorem the formula of the distance from one point x to a halfspace. Now, given $(x_1, \cdots, x_n)$ with labels $(y_1, \cdots, y_n)$, 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.

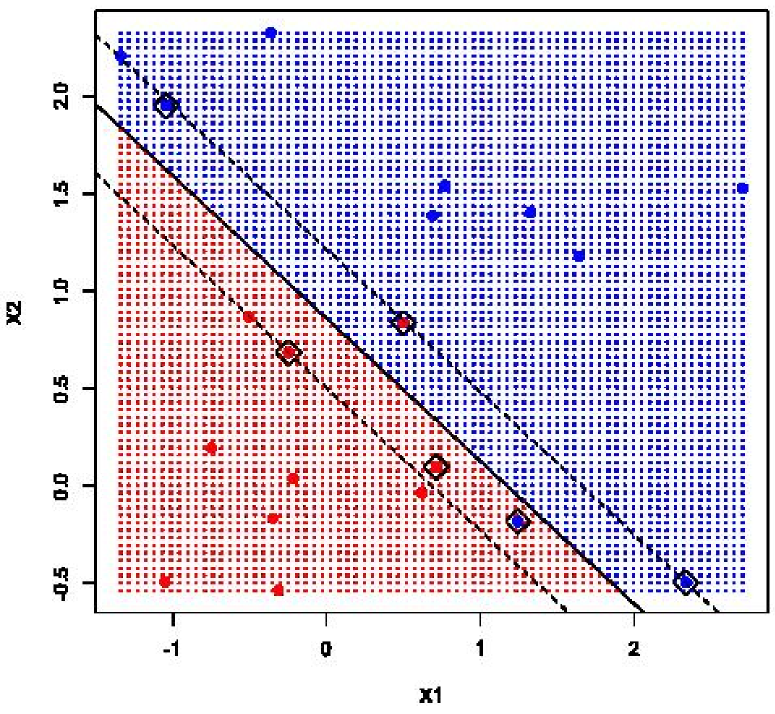

Figure 2.1: Linear SVM with the margin

The goal is to maximize the margin. This is solving by the following optimization problem:

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

This problem is equivalent to,

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

which is equivalent to,

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

or,

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

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

$$\begin{array}{lll} & \min_{\beta_0,\beta} & \|\beta\|^\circ \\ (2.6) & \text{subject to:} & \min_i \quad y_i(\beta_0+\beta^t x_i) = 1 \\ & & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array}$$

It is easily seen that this is equivalent to,

$$\begin{array}{lll} & \min_{\beta_0,\beta} & \|\beta\|^\circ \\ (2.7) & \text{subject to:} & \min_i \quad y_i(\beta_0+\beta^t x_i) \geq 1 \\ & & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array}$$

i.e.,

$$\begin{array}{lll} & \min_{\beta_0,\beta} & \|\beta\|^\circ \\ (2.8) & \text{subject to:} & y_i(\beta_0+\beta^t x_i) \geq 1, \ \forall\, i \in \mathcal{I} \\ & & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array}$$

In the Euclidean case we have:

$$\begin{array}{lll} & \min_{\beta_0,\beta} & \beta^t\beta \\ (2.9) & \text{subject to:} & y_i(\beta_0+\beta^t x_i) = 1, \ \forall\, i \in \mathcal{I} \\ & & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array}$$

which is an optimization problem with convex objective function and linear constraints. Then the problem (2.9) is equivalent to:

$$\begin{array}{lll} & \min_{\beta_0,\beta} & \beta^t\beta \\ (2.10) & \text{subject to:} & y_i(\beta_0+\beta^t x_i) > 0, \ \forall\, i \in \mathcal{I} \\ & & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array}$$

For polyhedral norms, problem (2.8) can be written as a linear problem. Let us consider the particular important cases $\|.\| = \|.\|_1$ and $\|.\| = \|.\|_\infty$. To achieve the dual of those norms we have the following property:

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

$$\tfrac{1}{p} + \tfrac{1}{q} = 1$$

If we have $\|.\| = \|.\|_1$, then its dual $\|.\|^\circ$ is the infinity norm $\|.\|_\infty$, and the problem (2.8) can be expressed as follows:

$$\begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|_\infty \\ \text{subject to:} & \min_i \;\; y_i(\beta_0 + \beta^t x_i) \geq 1, \;\; \forall \; i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R}. \end{array} \tag{2.11}$$

This problem can be reformulated as a linear problem,

$$\begin{array}{ll} \min & z \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \;\; \forall \; i \in \mathcal{I} \\ & z \geq \beta_i \geq -z \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R}, z \geq 0 \end{array} \tag{2.12}$$

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

$$\begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|_1 \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \;\; \forall \; i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \end{array} \tag{2.13}$$

which can also be converted in a linear problem,

$$\begin{array}{ll} \min & \sum_j z_j \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \;\; \forall \; i \in \mathcal{I} \\ & z_j \geq \beta_i \geq -z_j, \;\; j = 1,..,r \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R}, z_j \geq 0 \end{array} \tag{2.14}$$

2.1.2 The Linearly Nonseparable Case

In real applications, it is unlikely that there will be such a clear linear separation between data drawn from 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 so, or that no clear separability exists between the two classes, linearly or nonlinearly.

In the previous section we assumed that $\mathcal{I}$ was linearly separable, if this is not so the above problem is infeasible. Therefore, we must find other method.

One 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 sum in the objective function to control the perturbation. Hence, the following problem can be formulated:

$$\begin{array}{ll} \min & \beta^t\beta + C(\|\varepsilon\|_r)^r \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1, \forall\ i \in \mathcal{I}. \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall\ i \in \mathcal{I} \end{array} \tag{2.15}$$

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

$$\begin{array}{ll} \min & \beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1, \forall\ i \in \mathcal{I}. \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall\ i \in \mathcal{I} \end{array} \tag{2.16}$$

equivalent to,

$$\begin{array}{ll} \min & \frac{1}{2}\beta^t\beta + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) + \varepsilon_i \geq 1, \forall\ i \in \mathcal{I}. \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall\ i \in \mathcal{I} \end{array} \tag{2.17}$$

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

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

proceeding as the KKT method says,

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

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

$$\frac{\partial}{\partial\varepsilon_i}\mathcal{L}(\beta, \beta_0, \lambda, \eta, \varepsilon_i): \quad C - \lambda_i - \eta_i$$

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

$$\lambda = C - \eta_i$$

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

$$\begin{array}{lll} & \max & \sum_{i\in\mathcal{I}}\lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}}\lambda_i\lambda_j y_i y_j (x_i)^t x_j \\ (2.18) & \text{subject to:} & \sum_{i\in\mathcal{I}} y_i\lambda_i = 0,\ \lambda_i = C-\eta_i,\ \eta_i \geq 0 \\ & & \lambda_i \geq 0,\ \ \forall\ i\in\mathcal{I} \end{array}$$

We see that $0 \leq \lambda_i \leq C$, with $\lambda_i = C$ when $\varepsilon_i>0$ (which is when $y_i(\beta_0+\beta^t x_i) < 1$). Also when $y_i(\beta_0+\beta^t x_i) > 1$, $\varepsilon_i = 0$ since no cost is incurred, and $\lambda_i = 0$. When $y_i(\beta_0+\beta^t x_i) = 1$, λ_i can lie between 0 and 1, [15].

2.2 Nonlinear Support Vector Machine

At the beginning, we have discussed methods for constructing a linear SVM classifier. But if a linear classifier is not appropiate for the data learning set, can we extend the idea of linear SVM to the nonlinear case?

The key to constructing a nonlinear SVM is to observe that the observations in Ω only enter the dual optimization problem through the inner products $\langle x_i, x_j\rangle = x_i^t x_j,\ i,j = 1,2,..,n$.

In order to build a nonlinear SVM, we need some nonlinear tranformations, [16]. 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 $\langle .,.\rangle$ and norm $\|.\|$.

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,

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

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 transforme 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 aboved-mentioned problem of maximizing the margin can be reformulated now as:

$$\begin{array}{ll} \min & \|\beta\|^\circ \\ \text{subject to:} & y_i(\beta_0 + \beta^t\phi(x_i)) \geq 1, \forall\ i \in \mathcal{I} \\ & \beta \in \mathcal{F}, \beta_0 \in \mathbb{R} \end{array} \tag{2.19}$$

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

$$\begin{array}{ll} \min & \beta^t\beta \\ \text{subject to:} & y_i(\beta_0 + \beta^t\phi(x_i)) \geq 1, \forall\ i \in \mathcal{I} \\ & \beta \in \mathcal{F}, \beta_0 \in \mathbb{R}. \end{array} \tag{2.20}$$

This problem is equivalent to,

$$\begin{array}{ll} \min & \frac{1}{2}\beta^t\beta \\ \text{subject to:} & y_i(\beta_0 + \beta^t\phi(x_i)) \geq 1, \forall\ i \in \mathcal{I} \\ & \beta \in \mathcal{F}, \beta_0 \in \mathbb{R} \end{array} \tag{2.21}$$

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

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

We proceed as the method says:

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

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

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

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

$$\begin{array}{ll} \max & \sum_{i\in\mathcal{I}} \lambda_i - \frac{1}{2}\sum_{i,j\in\mathcal{I}} \lambda_i\lambda_j y_i y_j \phi(x_i)^t\phi(x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}} y_i\lambda_i = 0 \\ & \lambda_i \geq 0,\ \forall\ i \in \mathcal{I} \end{array} \tag{2.22}$$

2.2.1 The "Kernel Trick"

Remember that the idea behind nonlinear SVM is 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 appropiate.

At first, we would expect the dimensionality of $\mathcal{H}$ to be a huge impediment to constructing an optimal separating hyperplane, and a classification rule, because of the course of dimensionality. The fact that this does not become a problem in practice is due to the "Kernel Trick", which was first applied to SVM by Cortes and Vapnik in 1995, [9].

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.

2.2.2 Kernels and Their Properties

A kernel K is a function $K : \mathbb{R}^r \times \mathbb{R}^r \longrightarrow \mathbb{R}$, $\forall\ x_i, z_i \in \mathbb{R}^r$:

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

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 an example, the problem (2.22) can be reformulated using the "*Kernel Trick*" as:

$$\begin{array}{ll} \max & \sum_{i\in\mathcal{I}} \lambda_i - \frac{1}{2} \sum_{i,j\in\mathcal{I}} \lambda_i \lambda_j y_i y_j K(x_i, x_j) \\ \text{subject to:} & \sum_{i\in\mathcal{I}} y_i \lambda_i = 0 \\ & \lambda_i \geq 0, \quad \forall\ i \in \mathcal{I} \end{array} \tag{2.23}$$

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. We are going to see some kernels properties:

First, 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. Seeing the above result, 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. This result implies that the family of kernels is a convex cone. The multiplication of two kernels K_1 and K_2 yields a kernel

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

these above properties imply that any polynomial with possitive coefficients, $pol^+ = \{\sum_{i=1}^{n} \alpha_i x^i \mid n \in \mathbb{N},\ \alpha_1, .., \alpha_n \in \mathbb{R}^+\}$, 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. Next, 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. 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. Finally, 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 $f \in \mathcal{H}$,

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

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

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

Let $\{x_1, .., x_n\}$ be any set of n points in $\mathbb{R}^r$. Then, the $(n \times n)$-matrix $\mathbf{K} = ((K_{ij}))$, where $K_{ij} = K(x_i, x_j),\ i, j = 1, 2, .., n$, is called the *Gram* matrix of K with respect to $\{x_1, .., x_n\}$. 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 $\langle ., . \rangle_{\mathcal{H}_\mathcal{K}}$ and $\|.\|_{\mathcal{H}_\mathcal{K}}$, 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 $k(0) = 1$.

2.2.3 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.

Kernel	$K(x_i, z_i)$
Polynomial of degree d	$(\langle x_i, z_i \rangle + c)^d$
Gaussian radial basis function	$exp\{-\frac{\|x_i - z_i\|^2}{2\sigma^2}\}$
Laplacian	$exp\{\frac{\|x_i - z_i\|}{\sigma}\}$
Thin-plate spline	$(\frac{\|x_i - z_i\|}{\sigma})^2 log_e\{\frac{\|x_i - z_i\|}{\sigma}\}$
Sigmoid	$tanh(a\langle x_i, z_i \rangle + b)$

Table 2.1: Examples of Kernels

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

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

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,

$$K(x_i, z_i) = (\langle x_i, z_i \rangle + c)^2 = (x_1z_1 + x_2z_2 + c)^2 = \langle \phi(x_i), \phi(z_i) \rangle$$

where $\phi(x_i) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2c}x_1, \sqrt{2}x_2, c)^t$ 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. The dimensionality of $\mathcal{H}$ can rapidly become very large: for example, in visual recognition problems, data may consist of 16×16 pixel image, so that image is turned into a vector of dimension r =256. If $d = 2$, then $dim(\mathcal{H}) = 33670$, whereas if $d = 4$, we have $dim(\mathcal{H}) = 186043585$.

Other popular kernels are given by the Table 2.1. For example, the Gaussian RBF, Laplacian, and thin-plate spline kernels are example or stationary, or translation-invariant, kernels having the general form $K(x_i, z_i) = k(x_i - z_i)$, how we explained above. 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 hepful. If no such information is available, the most popular approach is to try either a Gaussian RBF, 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 be applied then to compare results.

2.2.4 Building the Optimal Path

The previous written problem (2.17) can be easily reformulated as:

$$\begin{array}{ll} \min & \frac{\mu}{2}\beta^t\beta + \sum_{i\in\mathcal{I}} \varepsilon_i \\ \text{subject to:} & 1 - y_i(\beta_0 + \beta^t x_i) \geq \varepsilon_i, \forall\ i \in \mathcal{I}. \\ & \beta \in \mathcal{F}, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0 \end{array} \tag{2.24}$$

where $\mu = \frac{1}{C}$. Its Lagrange primal function function is the same as in the problem (2.17):

$$\mathcal{L}(\beta, \beta_0, \lambda, \eta,\ \varepsilon_i) = \tfrac{\mu}{2}\beta^t\beta + \textstyle\sum_{i\in\mathcal{I}} \varepsilon_i - \sum_{i\in\mathcal{I}} \lambda_i\{y_i(\beta_0 + \beta^t x_i) - (1 - \varepsilon_i)\} - \sum_{i\in\mathcal{I}} \eta_i\varepsilon_i$$

Then, KKT constraints are the same too. We wish to find the entire solution path for all values of $\mu \geq 0$. The basic idea of our algorithm is as follows. We start with μ large and decrease it towards zero, keeping track of all the events that occur along the way. As μ decreases, $\|\beta\|$ increases, and hence the width of the margin decreases. As this width decreases, points move from being inside to outside the margin. Their corresponding λ_i

change from $\lambda_i = 1$ when they are inside the margin ($y_i(\beta_0 + \beta^t x_i) < 1$) to $\lambda_i = 0$ when they are outside the margin ($y_i(\beta_0 + \beta^t x_i) > 1$).

By continuity, points must linger on the margin ($y_i(\beta_0 + \beta^t x_i) = 1$) while their λ_i decrease from 1 to 0. We will see that the $\lambda_i(\mu)$ trajectories are piecewise-linear in μ, which affords a great computational saving: as long as we establish the break points, all values in between can be found by simple linear intepolation. Note that points can return to the margin, after having passed through it.

We can denote by H^+ the set of indices corresponding to $y_i = 1$ points, there being $n_+ = |H^+|$ in total. Likewise for H^- and n_-. This algorithm keeps track of the following sets, [15]:

$\mathcal{E} = \{\ i : y_i(\beta_0 + \beta^t x_i) = 1,\ 0 \leq \lambda_i \leq 1\ \}$, for Elbow

$\mathcal{L} = \{\ i : y_i(\beta_0 + \beta^t x_i) < 1,\ \lambda_i{=}1\ \}$, left of the Elbow

$\mathcal{R} = \{\ i : y_i(\beta_0 + \beta^t x_i) > 1,\ \lambda_i{=}0\ \}$, right of the Elbow

We are going to start with the initialization process. We need to establish the initial state of the sets defined above. When μ is very large, as ∞, from the KKT constraints we have $\beta = 0$, and the initial values of β_0 and the λ_i depend on whether $n_- = n_+$ or not. If the classes are balanced, one can directly find the initial configuration by finding the most extreme points in each class. We will see that when $n_- \neq n_+$, this is no longer the case, and in order to satisfy the KKT constraints, a quadratic programming algorithm is needed to obtain the initial configuration.

Case 1: $n_- = n_+$

Lemma 2.2.1 *For μ sufficiently large, all the $\lambda_i = 1$. The initial $\beta_0 \in [-1, 1]$ (any value gives the same loss $\sum_{i\in\mathcal{I}} \varepsilon_i = n_+ + n_-$).*

Proof. This proof relies on the criterion and the KKT conditions in the problem (2.24). Since $\beta = 0$, $f(x) = \beta_0$. To minimize $\sum_{i\in\mathcal{I}} \varepsilon_i$, we should clearly restrict β_0 to [-1,1], For $\beta_0 \in [-1, 1]$, all the $\varepsilon_i > 0$, $\eta_i = 0$ in the KKT constraint $\lambda_i = 1 - \eta_i$, and hence $\lambda_i = 1$. Picking one of the endpoints, say β_0= -1, causes $\lambda_i = 1$, $i \in H^+$, and hence also $\lambda_i = 1$, $i \in H^-$, for KKT constraint $\sum_{i\in\mathcal{I}} y_i\lambda_i = 0$ to hold. ■

We also have that for these early and large values of μ

$$\beta = \tfrac{1}{\mu}\beta^*, \text{ where } \beta^* = \textstyle\sum_{i\in\mathcal{I}} y_i x_i$$

Now in order that KKT constraints $\lambda_i = 1 - \eta_i$ remain satisfied, we need that one or more positive and negative instances hit the Elbow simultaneously. Hence as μ decreases, is required that $\forall\ i \in \mathcal{I},\ \ y_i(\beta_0 + \beta^t x_i) \leq 1$ or

$$y_i[\tfrac{(\beta^*)^t x_i}{\mu} + \beta_0] \leq 1 \tag{2.25}$$

or

$$\begin{aligned} &\beta_0 \leq 1 - \tfrac{(\beta^*)^t x_i}{\mu}, \forall\ i \in H^+ \\ &\beta_0 \geq -1 - \tfrac{(\beta^*)^t x_i}{\mu}, \forall\ i \in H^- \end{aligned} \tag{2.26}$$

Pick $i_+ = arg\max_{i \in H^+}(\beta^*)^t x_i$ and $i_- = arg\min_{i \in H^-}(\beta^*)^t x_i$. Then at this point of entry and beyond for a while we have $\lambda_{i_+} = \lambda_{i_-}$, and $(\beta_0 + \beta^t x_{i_+}) = 1$ and $(\beta_0 + \beta^t x_{i_-}) = -1$. This gives us two equations to solve for the initial point of entry μ_0 and β_0, with solutions

$$\mu_0 = \tfrac{(\beta^*)^t x_{i_+} - (\beta^*)^t x_{i_-}}{2} \tag{2.27}$$

$$\beta_0 = -(\tfrac{(\beta^*)^t x_{i_+} + (\beta^*)^t x_{i_-}}{(\beta^*)^t x_{i_+} - (\beta^*)^t x_{i_-}}) \tag{2.28}$$

Case 2: $n_+ > n_-$

In this case, when $\beta = 0$, the optimal choice for β_0 is 1, and the loss is $\sum_{i \in \mathcal{I}} \varepsilon_i = n_-$. However, is also required that $\lambda_i = 1 - \eta_i$ holds.

Lemma 2.2.2 *With $\beta^*(\lambda) = \sum_{i \in \mathcal{I}} y_i \lambda_i x_i$, let*

$$\begin{aligned} \{\lambda_i^*\} \ = \ & arg\min_\lambda \|\beta^*(\lambda)\|^2 \\ & s.t.\ \lambda_i \in [0,1]\ \forall\ i \in H^+, \\ & \lambda_i = 1\ \forall\ i \in H^-, \\ & \textstyle\sum_{i \in H^+} \lambda_i = n_- \end{aligned} \tag{2.29}$$

Then for some μ_0, we have that for all $\mu > \mu_0$, $\lambda_i = \lambda_i^*$, and $\beta = \frac{\beta^*}{\mu}$, with $\beta^* = \sum_{i \in \mathcal{I}} y_i \lambda_i^* x_i$

Proof. The Lagrange dual corresponding to

$$\mathcal{L}_\mathcal{P} = \tfrac{\mu}{2}\beta^t\beta + \textstyle\sum_{i \in \mathcal{I}} \varepsilon_i - \sum_{i \in \mathcal{I}} \lambda_i \{y_i(\beta_0 + \beta^t x_i) - (1 - \varepsilon_i)\} - \sum_{i \in \mathcal{I}} \eta_i \varepsilon_i$$

is obtained by substituting KKT constraints into this equation

$$\mathcal{L}_\mathcal{D} = \textstyle\sum_{i \in \mathcal{I}} \lambda_i \text{ - } \frac{1}{2\mu} \sum_{i,j \in \mathcal{I}} \lambda_i \lambda_j y_i y_j x_i x_j$$

Since we start with $\beta = 0$, $\beta_0 = 1$ all the H^- points are misclassified, and hence we will have $\lambda_i = 1$, $\forall i \in H^-$, and hence, from $\sum_{i \in \mathcal{I}} y_i \lambda_i = 0$, $\sum_{i \in \mathcal{I}} \lambda_i = 2n_-$. This latter sum will remain $2n_-$ for a while as β grows away from zero. This means that during this phase, the first term in the Lagrange dual is constant; the second term is equal to $\frac{1}{2\mu}\|\beta^*(\lambda)\|^2$, and since we maximize the dual, this proves the result. ■

We now establish the "*starting point*" μ_0 and β_0 when the λ_i start to change. Let β^* be the fixed coefficient direction corresponding to λ_i^*:

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

There are two possible scenarios:

1. There exist two or more elements in H^+ with $0 < \lambda_i^* < 1$

2. $\lambda_i^* \in \{0, 1\}$, $\forall\ i \in H^+$

Consider the first scenario, and suppose $\lambda_i^* \in (0, 1)$. Let $i_- = arg\min_{i \in H^-} (\beta^*)^t x_i$. Then since the point i_+ remains on the margin until an H_- point reaches its margin, we can find

$$\mu_0 = \frac{(\beta^*)^t x_{i_+} - (\beta^*)^t x_{i_-}}{2}$$

identical in form to (2.27), as is the corresponding β_0 to (2.28). For the second scenario, it is easy to see that we find ourselves in the same situation as in the case 1 (a point from H^- and one of the points in H^+ with $\lambda_i^* = 1$ must reach the margin simultaneously). Hence we get an analogous situation, except with $i_+ = arg\max_{i \in H_1^+} (\beta^*)^t x_i$, where H_1^+ is the subset of H^+ with $\lambda_i^* = 1$.

The Path.

The algorithm hinges on the set of points $\mathcal{E}$ sitting at the elbow of the loss function (i.e. on the margin). These points have $y_i(\beta_0 + \beta^t x_i) = 1$ and $\lambda_i \in$ [0,1]. These are distinct from the points $\mathcal{R}$ to the right of the elbow, with $y_i(\beta_0 + \beta^t x_i) > 1$ and $\lambda_i = 0$, and those points $\mathcal{L}$ to the left with $y_i(\beta_0 + \beta^t x_i) < 1$ and $\lambda_i = 1$. We consider this set at the point that an event occurred. The event can be either:

1. The initial event, which means 2 or more points start at the elbow, with their initial values of $\lambda \in$[0,1].

2. A point from $\mathcal{L}$ has just entered $\mathcal{E}$, with its values of λ_i initially 1.

3. A point from $\mathcal{R}$ has reentered $\mathcal{E}$, with its value of λ_i initially 0.

4. One or more points in $\mathcal{E}$ has left the set, to join either $\mathcal{R}$ or $\mathcal{L}$.

Whichever the case, for continuity reasons this set will stay stable until the next event occurs, since to pass through $\mathcal{E}$, a point λ_i must change from 0 to 1 or vice versa. Since all points in $\mathcal{E}$ have $y_i(\beta_0 + \beta^t x_i) = 1$, we can establish a path for their λ_i.

Event 4 allows for the possibility that $\mathcal{E}$ becomes empty while $\mathcal{L}$ is not. If this occurs, then the KKT condition $\sum_{i\in\mathcal{I}} y_i\lambda_i = 0$ implies that $\mathcal{L}$ is balanced w.r.t. +1's and -1's, and we resort to the initial condition as in case 1.

We use the subscript ℓ to index the sets above immediately after the ℓth event has occurred. Suppose $|\mathcal{E}_\ell| = m$, and let λ_i^ℓ, β_0^ℓ and μ_ℓ be the values of these parameters at the point of entry. Likewise f^ℓ is the function at this point. For convenience we define $\lambda_0 = \mu\beta_0$, and hence $\lambda_0^\ell = \mu_\ell\beta_0^\ell$. Since

$$f(x) = \tfrac{1}{\mu}\left(\textstyle\sum_{j\in\mathcal{I}} y_j\lambda_j K(x,x_j) + \lambda_0\right) \tag{2.30}$$

for $\mu_\ell > \mu > \mu_{\ell+1}$ we can write

$$\begin{aligned} f(x) &= [f(x) - \tfrac{\mu_\ell}{\mu} f^\ell(x)] + \tfrac{\mu_\ell}{\mu} f^\ell(x) = \\ &= \tfrac{1}{\mu}\left[\textstyle\sum_{j\in\mathcal{E}_\ell}(\lambda_j - \lambda_j^\ell) y_j K(x,x_j) + (\lambda_0 - \lambda_0^\ell) + \mu_\ell f^\ell(x)\right] \end{aligned} \tag{2.31}$$

The second line follows because all the observations in $\mathcal{L}_\ell$ have their $\lambda_i = 1$, and those in $\mathcal{R}_\ell$ have their $\lambda_i = 0$, for this range of μ. Since each of the m points $x_i \in \mathcal{E}_\ell$ are to stay at the elbow, we have that

$$\tfrac{1}{\mu}\left[\textstyle\sum_{j\in\mathcal{E}_\ell}(\lambda_j - \lambda_j^\ell) y_i y_j K(x,x_j) + y_i(\lambda_0 - \lambda_0^\ell) + \mu_\ell\right] = 1,\ \forall\ i \in \mathcal{E}_\ell \tag{2.32}$$

Writing $\delta_j = \lambda_j^\ell - \lambda_j$, from the above equation we have

$$\textstyle\sum_{j\in\mathcal{E}_\ell} \delta_j y_i y_j K(x_i,x_j) + y_i\delta_0 = \mu_\ell - \mu,\ \forall\ i \in \mathcal{E}_\ell \tag{2.33}$$

Futhermore, since at all times $\sum_{j\in\mathcal{I}} y_j\delta_j = 0$, we have that

$$\textstyle\sum_{j\in\mathcal{E}_\ell} y_j\delta_j = 0 \tag{2.34}$$

Equations (2.33) and (2.34) constitute $m+1$ linear equations in $m+1$ unknows δ_j, and can be solved. Denoting by K_ℓ^* the $m\times m$ matrix with i jth entry $y_i y_j K(x_i,x_j)$ for i and j in $\mathcal{E}_\ell$, we have from (2.33) that

$$K_\ell^*\delta + \delta_0 \underline{y}_\ell = (\mu_\ell - \mu)\underline{1} \tag{2.35}$$

where $\underline{y}_\ell$ is the m vector with entries y_i, $i \in \mathcal{E}_\ell$. From (2.34) we have

$$(\underline{y}_\ell)^t\delta = 0 \tag{2.36}$$

We can combine these two into one matrix equation as follows. Let

$$A_\ell = \begin{pmatrix} 0 & (\underline{y}_\ell)^t \\ \underline{y}_\ell & K_\ell^* \end{pmatrix},\ \delta^a = \begin{pmatrix} \delta_0 \\ \delta \end{pmatrix},\ \underline{1}^a = \begin{pmatrix} 0 \\ \underline{1} \end{pmatrix}. \tag{2.37}$$

Then (2.35) and (2.36) can be written

$$A_\ell \delta^a = (\mu_\ell - \mu)\underline{1}^a \tag{2.38}$$

If A_ℓ has full rank, then we can write

$$\underline{b}^a = (A_\ell)^{-1}\underline{1}^a \tag{2.39}$$

and hence

$$\lambda_j = \lambda_j^\ell - (\mu_\ell - \mu)b_j,\ j \in \{0\} \cup \mathcal{E}_\ell \tag{2.40}$$

Hence for $\mu_{\ell+1} < \mu < \mu_\ell$, the λ_j for points at the elbow proceeds linearly in μ. From (2.31) we have

$$f(x) = \tfrac{\mu_\ell}{\mu}[f^\ell(x) - h^\ell(x)] + h^\ell(x) \tag{2.41}$$

where

$$h^\ell(x) = \textstyle\sum_{j\in\mathcal{E}_\ell} y_j b_j K(x, x_j) + b_0 \tag{2.42}$$

Thus the function itself changes in a piecewise-inverse manner in μ. If A_ℓ does not have full rank, then the solution paths for some of the λ_i are not unique, and more care has to be taken in solving the system (2.38). This occurs, for example, when two training observations are identical. Other degeneracies can occur, but rarely in practice, such as three different points on the same margin in $\mathbb{R}^2$.

Finding $\mu_{\ell+1}$

The paths (2.40) and (2.41) continue until one of the following events occur:

1. One of the λ_i for $i \in \mathcal{E}_\ell$ reaches a boundary (0 or 1). For each i the value of μ for which this occurs is easily established from (2.40).

2. One of the points in $\mathcal{L}^\ell$ or $\mathcal{R}^\ell$ attains $y_i(\beta_0 + \beta^t x_i) = 1$. From (2.41) this occurs for point i at

$$\mu = \mu_\ell\left(\frac{f^\ell(x_i) - h^\ell(x_i)}{y_i - h^\ell(x_i)}\right) \tag{2.43}$$

By examining these conditions, we can establish the largest $\mu < \mu_\ell$ for which an event occurs, and hence establish $\mu_{\ell+1}$ and update the sets. One special case not addressed above is when the set $\mathcal{E}$ becomes empty during the course of the algorithm. In this case, we revert to an initialization setup using the points in $\mathcal{L}$. It must be the case that these points have an equal number of +1's as -1's, and so we are in the balanced situation as in case 1: $n_+ = n_-$.

It is discussed in [15] that, by examining in detail the linear boundary in instances where $\rho = 2$, several different types of behavior can be observed:

1. If $|\mathcal{E}| = 0$, as μ decreases, the orientation of the decision boundary stays fixed, but the margin width narrows as μ decreases.

2. If $|\mathcal{E}|$=1 or $|\mathcal{E}|$=2, but with the pair of points of opposite classes, then the orientation typically rotates as the margin width gets narrower.

3. If $|\mathcal{E}|$=2, with both points having the same class, then the orientation reamains fixed, with the one margin stuck on the two points as the decision boundary gets shrunk toward in it.

4. If $|\mathcal{E}| \geq$3, then the margins and hence $f(x)$ remains fixed, as the $\lambda_i(\mu)$ change. This implies that $h^\ell = f^\ell$ in (2.41).

Termination.

In the separable case, we terminate when $\mathcal{L}$ becomes empty. At this point, all the ε_i in (2.17) are zero, and futher movement increases the norm of β unnecessarily.

In the non-separable case, μ runs all the way down to zero. For this to happen withput f "*blowing up*" in (2.41), we must have $f^\ell - h^\ell$ =0, and hence the boundary and margins remain fixed at a point where $\sum_{i\in\mathcal{I}} \varepsilon_i$ is as small as possible, and the margin is as wide as possible subject to this constraints.

Computational Complexity.

At any update event ℓ along the path of the algorithm, the main computational burden is solving the system of equations of size $m_\ell = |\mathcal{E}_\ell|$. While this normally involves $O(m_\ell^3)$ computations, since $\mathcal{E}_{\ell+\infty}$ differs from $\mathcal{E}_\ell$ by typically one observation, inverse updating/downdating can reduce the computations to $O(m_\ell^2)$. The computation of $h^\ell(x_i)$ in (2.42) requires $O(nm_\ell)$ computations. Beyond that, several checks of cost $O(n)$ are needed to evaluate the next move.

2.3 SVM in R

We need to talk about some R libraries which help us to implement SVM in R. To begin with, we expose the library "*e*1071". In order to implement SVM in R, this library has three different commands, which do differents things, but related. When we explain these commands we will work with the Breast Cancer Wisconsin (BCW) dataset, [1]. This dataset has 569 instances and 32 attributes (ID, diagnosis and 30 real-valued input features). ID is the ID number and diagnosis has two classes: B-bening or M-malign. Ten real-valued features are computed for each cell nucleus:

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

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

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

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

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

i) symmetry

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

The mean, standard error, and worst or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius. This dataset does not have missing attribute values and his class distribution is: 357 benign and 212 malign. As a result of a principal components analysis we have the following image, with the benign and malign cases:

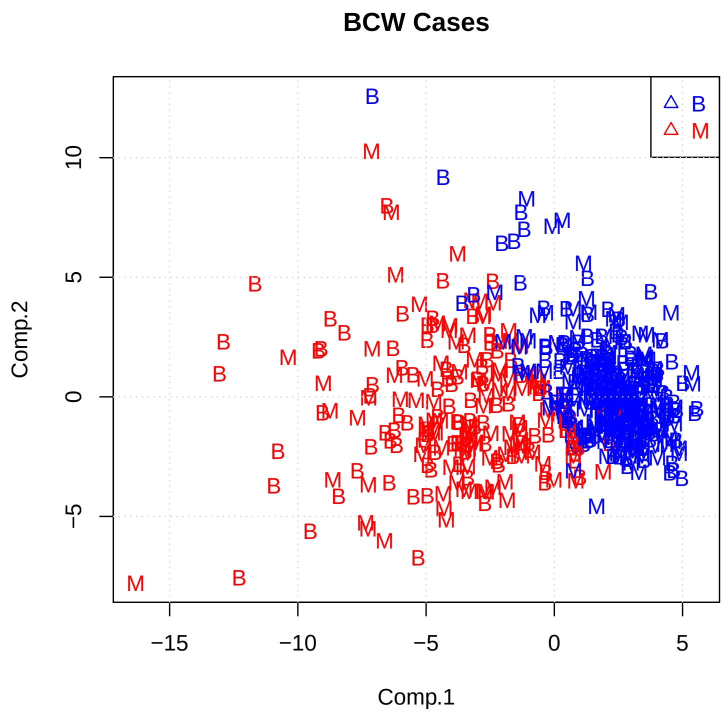

Figure 2.2: Representation of cases

In this image we can see that there are bening and malign cases with a wrong prediction. In the image, blue cases are the cases that are benign as predicted, and red cases are cases that according to prediction are malign. The cases with a B in the image are truly benign and the cases with an M in the image are truly malign. We are going to proceed to see the commands, used in R:

Support Vector Machines:

The command is called "*svm*". We are going to see the use of this command "*svm*":

```
## 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)
```

This command is used to train a support vector machine. 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.

An object of class "*svm*" contains the fitted model, including:

$SV \rightarrow$ the resulting support vectors

$index \rightarrow$ the index of the resulting support vectors in the data matrix. This index refers to the preprocessed data.

$coefs \rightarrow$ the corresponding coefficients times the training labels.

$rho \rightarrow$ the negative intercept.

$sigma \rightarrow$ in case of a probabilistic regression model, the scale parameter of the hypothesized laplace distribution estimated by maximum likelihood.

$probA$, $probB \rightarrow$ numeric vectors of length $\frac{k(k-1)}{2}, k$ number of classes, containing the parameters of the logistic distributions fitted to the decision values of the binary classifiers $(\frac{1}{(1+exp(ax+b)})$.

We can see an example with the Breast Cancer Wisconsin (BCW) dataset. First, we install the package 'e1071' and download this library:

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

After that, we download the dataset and delete the 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]
```

Define index as the *datos*' number of rows. Then, we divide the information between testset and trainset. Test set has the 30 % of the information and train set has the 70 %:

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

```
testset <- datos[testindex,]
trainset <- datos[-testindex,]
```

We can use a generic function called "*tune.svm*". This function tunes hyperparameters of 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)
```

which output is as follows:

```
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
```

Names of the variables can be seen with `names(datos)`:

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

As we said, variable V1 was named ID and was deleted. To conclude, we implement the svm model with the radial kernel:

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

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 Support Vector Machine:

In order to do this, the command is called "*predict.svm*". As the previous command, we apply this command "*predict*" on an object of class "*svm*":

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

This function predicts values based upon 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 names of the columns of the matrix indicate the labels of the two classes. If *probability* is TRUE, the vector gets "*probabilities*" attribute containing a $n \times k$ matrix of the class probabilities.

Some tools are used on the command:

object $\rightarrow$ describes an object of class "*svm*", created by command "*svm*".

newdata $\rightarrow$ 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 $\rightarrow$ is a logical value controlling whether the decision values of al binary classifiers computed in multiclass classification shall be computed and returned.

probability $\rightarrow$ is a logical value indicating whether class probabilities should be computed and returned. Only possible if the model was fitted with the probability option enabled.

na.action $\rightarrow$ 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.

We go on using the example before used in order to implement a example of "*predict.svm*" command. Now we run the model again with the test set in order to predict classes:

```
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 is:

```
      true
    pred B M
     B 110 5
     M 0 55
```

This means that there are 110 benign instances in test set and all of them were predicted as benign instances. On the other hand, there are 60 malign instances in test set, 55 were predicted rightly and 5 as benign instances.

Let:

TP: true positive, i.e. malign instances predicted rightly
FP: false positive, i.e. benign instances predicted as malign
TN: true negative, i.e. benign instances predicted rightly
$|N|$: total of benign instances
$|P|$: total of malign instances

Sensitivity=$\frac{TP}{|P|}$

Specificity=$\frac{TN}{|N|}$

Precision=$\frac{TP}{TP+FP}$

For this problem we have:

Sensitivity=$\frac{55}{60} = 0.916$

Specificity=$\frac{110}{110} = 1$

Precision=$\frac{55}{55+0} = 1$

Plot SVM Objects:

In order to do this, the command is called "*plot.svm*", where we simply apply the command "*plot*" on a class "*svm*":

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

It is the use of the command "*plot.svm*", which have the following arguments:

x $\rightarrow$ object of class "*svm*".

data $\rightarrow$ the data which we will visualize. Should be the same used for fitting.

$formula$ $\rightarrow$ the formula selects the visualized two dimensions.

$fill$ $\rightarrow$ a switch which indicates whether a contour plot for the class regions should be added.

slice $\rightarrow$ only is needed if more than two variables are used. Is a list of named values for the dimension held constant.

symbolPalette $\rightarrow$ the color palette used for the class the data points and support vectors belong to.

svSymbol $\rightarrow$ symbols used for support vectors.

dataSymbol $\rightarrow$ symbols used for data points.

We can use additional graphics parameters. Command "*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, draws a filled contour plot of the class regions. We can see some examples:

```
## a simple example
data(cats, package = "MASS")
m <- svm(Sex ., data = cats)
plot(m, cats)

## more than two variables:  fix 2 dimensions
data(iris)
m2 <- svm(Species ., data = iris)
plot(m2, iris, Petal.Width   Petal.Length,
slice = list(Sepal.Width = 3, Sepal.Length = 4))

## plot with custom symbols and colors
plot(m, cats, svSymbol = 1, dataSymbol = 2, symbolPalette = rainbow(4),
color.palette = terrain.colors)
```

Another library is called "*SvmPath*". Related with the above section, the "*SvmPath*" algorithm can be started at any intermediate solution of the SVM optimization problem (i.e. the solution for any μ), since the values of λ_i and $f(x_i)$ determinate the sets $\mathcal{L}$, $\mathcal{E}$ and $\mathcal{R}$. We are going to see some commands of the library "*SvmPath*":

Fit the entire regularization path for a 2-class SVM:

The command is called "*svmpath*". A "*svmpath*" object is returned, for which there are print, summary, coef and predict methods. The SVM has a regularization or cost parameter C, which controls the amount by which points overlap their soft margins. Typically either a default large value for C is chosen (allowing minimal overlap), or else a few values are compared using a validation set. This algorithm computes the entire regularization path (i.e. for all possible values of C for which the solution changes), with a cost a small ($\sim$3) multiple of the cost of fitting a single model. The command is used as follows:

```
svmpath(x, y, K, kernel.function = poly.kernel, param.kernel = 1, trace,
plot.it, eps = 1e-10, Nmoves)
```

Some arguments are used:

x $\rightarrow$ the data matrix ($n \times p$) with n rows (observations) on p variables (columns)

y $\rightarrow$ the "{-1,+1}" valued response variable

K $\rightarrow$ an $n \times n$ kernel matrix, with default value K= $kernel.function(x, x)$

$kernel.function$ $\rightarrow$ this is a user-defined function. Provided are *poly.kernel* (the default, with parameter set to default to a linear kernel) and *radial.kernel*

param.kernel $\rightarrow$ parameter(s) of the kernels

trace $\rightarrow$ if TRUE, a progress report is printed as the algorithm runs; default is FALSE

plot.it $\rightarrow$ a flag which indicates whether a plot should be produced (default FALSE; only usable with p=2

eps $\rightarrow$ a small machine number which is used to identify minimal step sizes

$Nmoves$ $\rightarrow$ the maximum number of moves

Additional arguments to some of the functions called by "*svmpath*" can be used. One such argument that can be passed is *ridge*. This is used to produce stable solutions to linear equations. The algorithm used in `svmpath()` is described in detail in "The Entire Regularization Path for the Support Vector Machine" by Hastie, Rosset, Tibshirani and Zhu, [15]. It exploits the fact that the hinge loss-function is piecewise linear, and the penalty term is quadratic. This means that in the dual space, the lagrange multipliers will be piecewise linear.

Recursive Feature Elimination (RFE):

Another package can be useful: "*pathClass*". This package contains an implementation of the Recursive Feature Elimination (RFE), which we will see in the Chapter 3. The used command is the "$fit.rfe$" command. The command is used as follows:

```
fit.rfe(x, y, DEBUG = FALSE,
scale = c("center", "scale"), Cs = 10∧c(-3:3), stepsize = 0.1)
```

This command has different type of arguments:

x $\rightarrow$ a $p \times n$ matrix of expression measurements with p samples and n genes.

$y \to$ a factor of length p comprising the class labels.

$DEBUG$ $\to$ debugs the information which has to be plotted.

scale $\to$ a character vector which defines if the data should be centered and/or scaled. Possible values are center and/or scale. Defaults to c(!center!, !scale!).

Cs $\to$ soft-margin tuning parameter of the SVM. Defaults to 10∧c(-3:3).

stepsize $\to$ amount of features that are discarded in each step of the feature elimination. Defaults to 10%.

The output of this command is a RFE fit object. $features$ = selected features, *error.bound*= span bound of the model and fit = fitted SVM model. In the Chapter 3 we will see an example of the application of the RFE algorithm to the Breast Cancer Wisconsin dataset in R.

2.4 Ramp Loss SVM

Traditional SVM with the linear kernel is often used but performance is diminished with the presence of outliers. Carrizosa and Romero Morales, [7], provide an extensive review of the relationship between mathematical optimization and classification, including many approches for SVM. Below we review some results relating to ramp loss SVM.

Ramp loss SVM, [25, 27, 2, 6], presents a solution to the outlier problem with formulations of SVM that gives less weight to problematic training points and therefore better generalizability as compared to traditional SVM. Although performance is greatly improved, these formulations are computationally intractable with this approach. To date, computational studies of the robustness of ramp loss SVM are limited. Error measurement for SVM with ramp loss differs from traditional SVM in that all misclassified observations falling outside of the margin add a loss of 2 to the objective function while the error for observations falling in the margin is between 0 and 2, and depends on the distance to the margin boundary, [6]. Shen at al.,[25], Wang et al., [27], and Collobert et al., [8], suggest algorithms for ramp loss SVM that converge to locally optimal solutions.

Related efforts to increase robustness to outlier observations include discrete SVM (DSVM), [20, 21, 22], that implements SVM with the hard margin loss and the linear kernel. The hard margin loss assigns an error of one to observations between the margin boundaries and those outside the margin boundaries and misclassified, and zero otherwise. DSVM is formulated as a mixed integer linear program (MILP). Utsun et al., [26], formulate an optimization-based linear classification method that penalizes the number of misclassifications and both the ℓ_1 and the number of nonzero coefficients of the discriminant via MILP.

Traditional SVM uses regularization to avoid overfitting noise by introducing a quadratic term in the objective function of the corresponding optimization formulation. Finding an optimal SVM hyperplane for a given training problem dataset requires the solution of a quadratic problem. By linearizing the regularization term, the training problem becomes a linear program (LP). Robust LP formulations for finding optimal hyperplanes for discrimination of linearly inseparable sets were studied in the early 1990's. Bennett and Mangasarian, [3], proposed a single LP formulation which generates a plane that minimizates the average errors of misclassified points belonging to two disjointed sets of observations in n-dimensional real space. Mangasarian, [19], introduced a generalized SVM formulation that could be used for both quadratic and LP formulations. Hadzic, [14], and Kecman, [17], provide a formulation that utilizes the ℓ_1-norm of the separating hyperplane for maximizing the margin. It performed well in several empirical tests, [17, 14]. Zhou et al., [29], introduce a LP SVM formulation in which the margin is defined as the right hand side of the constraint, $y_i(\beta_0 + \beta^t x_i) \geq 1$, with an unknown in place of 1. This is unique because the right hand side of these constraints is representative of the margin width of the separating hyperplanes.

We assume training data are given consisting, as we saw, of observations $x_i \in \mathbb{R}^r,\ i = 1, \cdots, n$ each having an associated class label $y_i \in \{-1, 1\}$.

To produce a ramp loss function the errors are weighted as follows:

$$R(x_i, f(y_i, f(x_i))) = \begin{cases} 0, & y_i f(x_i) > 1, \\ 1 - y_i f(x_i), & -1 \leq y_i(x_i) \leq 1, \\ 2, & y_i f(x_i) \leq 1. \end{cases}$$

Brooks, [6], presents MIQP formulations for SVM with ramp loss error terms. The formulation called SVMIP1-RL can be used to find a hyperplane that maximizes the margin while minimizing the ramp loss:

$$\text{(2.44)} \quad \begin{array}{lll} \text{(SVMIP1-RL)} & \min_{\beta,\beta_0,\varepsilon,z} & \{\frac{1}{2}\|\beta\|^2 + C\sum_{i\in\mathcal{I}}(\varepsilon_i + 2z_i)\} \\ & \text{subject to:} & y_i(\beta_0 + \beta^t x_i) = 1 - \varepsilon_i \text{ if } z_i = 0, \quad \forall\, i \in \mathcal{I}, \\ & & 0 \leq \varepsilon_i \leq 2, \quad \forall\, i \in \mathcal{I}, \\ & & z_i \in \{0, 1\}, \quad \forall\, i \in \mathcal{I}. \end{array}$$

Just like traditional SVM, this ramp loss model can generate nonlinear discriminants by substituting primal variables with dual variables; i.e., $\beta = \sum_{i=1}^n \lambda_i y_i x_i$, and replacing x_i with $\phi(x_i)$. Replacement of these variables, as well as application of the kernel trick, i.e. $x_i.x_j \rightarrow \phi(x_i).\phi(x_j) = K(x_i, x_j)$ gives rise to the following model:

(2.45)

$$\begin{array}{lll} \text{(SVMIP2-RL)} & \min_{\lambda,\beta_0,\varepsilon,z} & \{\frac{1}{2}\sum_{i,j\in\mathcal{I}}\lambda_i\lambda_j y_i y_j K(x_i, x_j) + C\sum_{i\in\mathcal{I}}(\varepsilon_i + 2z_i)\} \\ & \text{subject to:} & y_i(\beta_0 + \sum_{j\in\mathcal{I}}\lambda_j y_j K(x_i, x_j)) = 1 - \varepsilon_i \text{ if } z_i = 0, \quad \forall\, i \in \mathcal{I}, \\ & & 0 \leq \lambda_i \leq C, \quad \forall\, i \in \mathcal{I}, \\ & & 0 \leq \varepsilon_i \leq 2, \quad \forall\, i \in \mathcal{I}, \\ & & z_i \in \{0, 1\}, \quad \forall\, i \in \mathcal{I}. \end{array}$$

The conditional constraints can be used in a branch and bound solver directly via "indicator constraints". Alternatively, one can linearize the conditional constraints by replacing the right hand side with $1-\varepsilon_i - Mz_i$ for all training vectors. The choice of the parameter M is important because making it too small may bias the resulting discriminant towards outliers. On the other hand making M too large may lead to numerical instability. An option for controlling the size of M may be to solve the linear problem for the magnitudes of the ε's, and adjusting M as necessary.

Now we are going to combine ideas for ℓ_1-norm regularization with ramp loss SVM to produce four formulations, recently proposed in [6], that may be solved as MILPs. For each formulation for ℓ_1-norm regularization, incorporating the ramp loss amounts to either using conditional "indicator" constraints or adding a binary integer variable to each soft margin constraint.

Mangasarian, [18], introduced a generalized SVM formulation that considers functions for regularization that lead to quadratic programming and LP formulations for SVM and also allows for weighting the misclassification of observations. Two different options for an LP SVM are given. The first is the ℓ_1-norm of the dual variables, λ. This formulation can incorporate the ramp loss as follows:

(2.46)

$$\begin{array}{lll} \text{(GSVM1-RL)} & \min_{\lambda,s,\beta_0,\varepsilon,z} & \{\sum_{i\in\mathcal{I}} s_i + C\sum_{i\in\mathcal{I}}(\varepsilon_i + 2z_i)\} \\ & \text{subject to:} & y_i(\beta_0 + \sum_{j\in\mathcal{I}}\lambda_j y_j K(x_i,x_j)) \geq 1-\varepsilon_i + Mz_i, \quad \forall\, i \in \mathcal{I}, \\ & & s_i \leq \lambda_i \leq -s_i, \quad \forall\, i\in\mathcal{I}, \\ & & \varepsilon_i \geq 0, \quad \forall\, i \in \mathcal{I}, \\ & & z_i \in \{0,1\}, \quad \forall\, i\in\mathcal{I}. \end{array}$$

The variable s_i is the absolute value of λ_i. The second LP SVM uses absolute value of $\sum_{j\in\mathcal{I}}\lambda_j y_j K(x_i,x_j)$. The extension to incorporate the ramp loss is given in the following formulation:

(2.47)

$$\begin{array}{lll} \text{(GSVM2-RL)} & \min_{\lambda,s,\beta_0,\varepsilon,z} & \{\sum_{i\in\mathcal{I}} s_i + C\sum_{i\in\mathcal{I}}(\varepsilon_i + 2z_i)\} \\ & \text{subject to:} & y_i(\beta_0 + \sum_{j\in\mathcal{I}}\lambda_j y_j K(x_i,x_j)) \geq 1-\varepsilon_i + Mz_i, \quad \forall\, i \in \mathcal{I}, \\ & & s_i \leq \sum_{j\in\mathcal{I}}\lambda_j y_j K(x_i,x_j) \leq -s_i, \quad \forall\, i\in\mathcal{I}, \\ & & \varepsilon_i \geq 0, \quad \forall\, i \in \mathcal{I}, \\ & & z_i \in \{0,1\}, \quad \forall\, i\in\mathcal{I}. \end{array}$$

In examining (2.47), the first set of constraints can be seen as generalizing the soft margin constraints on each training observation in traditional SVM. The second set of constraints ensures that each s_i is the absolute value of the each regularization term. If the kernel is linear then it is interesting to note that the regularization term we are minimizing is $\sum_{i\in\mathcal{I}}|(\beta x_i)|$. In this way, the function that we are minimizing is perhaps more similar to traditional SVM than (2.46).

Hadzic, [14], and Kecman, [17], provide a formulation that utilizes the ℓ_1-norm of the separating hyperplane for maximizing the margin. The first formulation is derived from classical SVM with the only exception being that the ℓ_1-norm is used as a regularization term, instead of the ℓ_2-norm of the discriminant coefficients. Kecman and Hadzic's LP classifier can be modified to accommodate the ramp loss as follows:

(2.48)

$$\begin{array}{llll} \text{(MILPSVM-RL)} & \min_{\beta^+,\beta^-,\beta_0,\varepsilon} & \{\sum_{i\in\mathcal{I}}(\beta_i^+ + \beta_i^-) + \sum_{i\in\mathcal{I}}(\varepsilon_i + 2z_i)\} & \\ & \text{subject to:} & y_i(\beta_0 + x_i\beta^+ - x_i\beta^-) \geq 1 - \varepsilon_i + Mz_i, & \forall\, i \in \mathcal{I}, \\ & & \varepsilon_i \geq 0, & \forall\, i \in \mathcal{I}, \\ & & z_i \in \{0,1\}, & \forall\, i \in \mathcal{I}, \\ & & \beta^+, \beta^- \geq 0 & . \end{array}$$

In this formulation, the attribute vector β is split into positive β^+ and negative β^- parts so that the generalization terms in the objective function are linear instead of quadratic. This new measure is the ℓ_1-norm because the sum of β^+ and β^- equates to the sum of the absolute value of all parts of the vector β. This formulation is the closest literal transition from the traditional ℓ_2-norm regularization of SVM to ℓ_1-norm with ramp loss error term. However, it is not clear how the formulation can be used with the "kernel trick", and so only linear discriminants can be learned.

The fourth and last formulation for ramp loss SVM with ℓ_1 regularization is an extension of a LP formulation due to Zhou et al.[29]. They introduce a LP SVM formulation by considering the ℓ_∞-norm for regularizaion. Their formulation replaces the metric representing the magnitude, of the classifier margin in the objective function with a variable r in the equation for the margin, $d = \frac{2r}{\|\beta\|_\infty}$. By replacing the right hand side of the bounding constraints in traditional SVM with r (in place of 1), we allow for the substitution of terms in the objective function:

(2.49)

$$\begin{array}{llll} \text{(rSVM2-RL)} & \min_{\lambda,r,\beta_0,\varepsilon} & \{-r + C\sum_{i\in\mathcal{I}}(\varepsilon_i + 2z_i)\} & \\ & \text{subject to:} & y_i(\beta_0 + \sum_{j\in\mathcal{I}}\lambda_j y_j K(x_i,x_j)) \geq r - \varepsilon_i + Mz_i, & \forall\, i \in \mathcal{I}, \\ & & -1 \leq \lambda_i \leq 1, & \forall\, i \in \mathcal{I}, \\ & & 0 \leq \varepsilon_i \leq 2, & \forall\, i \in \mathcal{I}, \\ & & z_i \in \{0,1\}, & \forall\, i \in \mathcal{I}, \\ & & r \geq 0 & . \end{array}$$

Bounds are placed on λ, so that the absolute value of each variable is at most 1. Even with these constraints, however, the original LP formulation is unbounded when $C < \max\{\frac{1}{n^+}, \frac{1}{n^-}\}$, where n^+ and n^- represent the number of observations in each class. Also, setting all variables to zero always provides a feasible solution. Therefore, non-trivial solutions occur only when a feasible solution exists where the objective function value is negative. Therefore, C must be large enough to avoid an unbounded problem but small enough to avoid a rule that places all observations in one class. With the incorporation of the ramp loss, (2.49) is no longer unbounded. However, setting all variables to zero remains feasible with a zero objective function value.

Chapter 3

Feature Selection in SVM

As of 1997, when a special issue on relevance including several on variable and feature selection was published (Blum and Langley, 1997, Kohavi and John, 1997), few domains explored more than 40 features. The situation has changed since then, many papers explore domains with hundreds to tens of thousands of variables or features. New techniques are proposed to address these challenging tasks involving many irrelevant and redundant variables.

One example is typical of the new application domains and serves us as illustration about this problem. This one is gene selection from microarray data. In this problem, [11], the variables are gene expression coefficients corresponding to the abundance of mRNA in a sample, for a number of patients.

A typical classification task is to separate healthy patients from cancer patients, based on their gene expression profile. Usually fewer than 100 patients are available altogether for training and testing. But, the number of variables in the raw data ranges from 6000 to 60000. Some initial filtering usually brings the number of variables to a few thousand. Because the abundance of mRNA varies by several orders of magnitude depending on the gene, the variables are usually standardized.

Pattern classification, especially in the context of regularization that enforces sparsity of the weight vector, is deeply connected to the problem of feature selection. In the feature selection problem one would like to select a subset of features while preserving or improving the discriminative ability of a classifier. In many supervised learning problems feature selection is important for a variety of reasons: generalization performance, running time requirements, and constraints and interpretational issues imposed by the problem itself.

Some methods put more emphasis on one aspect than another. Some papers about this problem focus mainly on constructing and selection subsets of features that are useful to build a good predictor.

This contrasts with the problem of finding and ranking all potentially relevant variables.

Selecting the most relevant variables is usually suboptimal for building a predictor, particulary if the variables are redundant. Conversely, a subset of useful variables may exclude many redundant, but relevant, variables.

We need to talk about the three principal approaches of feature selection: Filters, Wrappers and Embedded methods, [12]:

Filters: Filter type methods select variables regardless of the model. They are based only on general features like the correlation with the variable to predict. Filter methods suppress the least interesting variables. The others variables will be part of the model classification, regression is used to classify the data prediction. These methods are particularly effective in computation time and robust to overfitting. However, filter methods tend to select redundant variables because they does not consider the relationships between variables. Therefore, they are mainly used as a pre-process method.

Wrappers: Wrappers methods evaluate subsets of variables which allows, unlike filter approaches, to detect the possible interactions between variables. This method have two disadvantages:

1. The increasing overfitting risk when the number of observations is insufficient.

2. The significant computation time when the number of variables is large.

Filters and wrappers differ mostly by the evaluation criterion, but both methods can make use of search strategies to explore the space of all possible feature combinations, that is usually too large to be explored exhaustively. It is usually understood that filters use criteria not involving any learning machine, e.g., a relevance index based on correlation coefficients or test stadistics, whereas wrappers use the performance of a learning machine trained using a given feature subset.

We need to talk about the Recursive Feature Elimination (RFE), [28]. RFE is a feature selection algorithm described by Guyon et al., [13]. The method, given that one wishes to employ only $r < n$ input dimensions in the final decision rule, attempts to find the best subset r. The method operates by trying to choose the r features which lead to the largest margin of class separation, using SVM classifier.

This combinatorial problem is solved in a greedy fashion as each iteration of training by removing the input dimension that decreases the margin the least until only r input dimensions remain, this is known as backward selection.

For SVMs, $W^2(\lambda) = \sum \lambda_i \lambda_j y_i y_j K(x_i, x_j)$ is a measure of predictive ability, and is inversely proportionate to the margin. The algorithm is thus to remove features which keep this quantity small. This can be done with the following iterative procedure:

$\rightarrow$ Given solution λ, calculate for each feature p:

$$W^2_{(-p)}(\lambda) = \sum \lambda_i \lambda_j y_i y_j K(x_i^{-p}, x_j^{-p})$$

where x_i^{-p} means training point i with feature p removed.

$\rightarrow$ Remove the feature with smallest value of $|W^2(\lambda) - W^2_{(-p)}(\lambda)|$.

If the classifier is a linear one (of type $f(x) = \beta^t x + \beta_0$), this algorithm corresponds to removing the smallest corresponding value of $|\beta_i|$ in each iteration. To speed up computations, when the number of features is large the author, [13], suggests to remove half of the features each iteration. Note that RFE has been designed for two-class problems although a multi-class version can be derived easily for a one-againts-the-rest approach. The idea is then to remove the features that lead to the smallest value of $\sum_{i=1}^{Q} |W_k^2(\lambda^k) - W^2_{k,(-p)}(\lambda^k)|$ where $W_k^2(\lambda)$ is the corresponding margin based value $W^2(\lambda^k)$ for the machine discriminating class k from all the others.

SVM-RFE is an application of RFE using the weight magnitude as ranking criterion. We present below an outline algorithm in the linear case. The algorithm can be generalized to remove more than one feature per step for speed reasons.

Algorithm SVM-RFE:

```
Inputs
Training examples
        $X_0 = [x_1, x_2, \cdots, x_k, \cdots, x_\ell]^t$
Class labels
        $\underline{y} = [y_1, y_2, \cdots, y_k, \cdots, y_\ell]^t$
Initialize:
Subset of surviving features
        $\underline{s} = [1, 2, \cdots, n]$
Feature ranked list
        $r = [\ ]$
Repeat until $\underline{s} = [\ ]$
Restrict training examples to good feature indices
        $X = X_0(:, \underline{s})$
Train a classifier
        $\alpha = SVM - train(X, \underline{y})$
Compute the weight vector of dimension length ($\underline{s}$)
        $w = \sum_k \alpha_k y_k x_k$
```

```
Compute the ranking criteria
        $c_i = (w_i)^2, \ \forall \ i$
Find the feature with smallest ranking criterion
        $f = arg\min(\underline{c})$
Update feature ranked list
        $\underline{r} = [\underline{s}(f), \underline{r}]$
Eliminate the feature with the smallest ranking criterion
        $\underline{s} = \underline{s}(1 : f - 1, f + 1 : length(\underline{s}))$
Output:
Feature ranked list $\underline{r}$.
```

In R we can run an explicit example of the SVM-RFE application. We use the Breast Cancer Wisconsin dataset. With the followings commands we obtain explicitly the SVM-RFE algorithm, with the help of the "*svm*" command:

```
svmrfeFeatureRanking = function(x,y){
n = ncol(x)

survivingFeaturesIndexes = seq(1:n)
featureRankedList = vector(length=n)
rankedFeatureIndex = n

while(length(survivingFeaturesIndexes)>0){
#train the support vector machine
svmModel = svm(x[, survivingFeaturesIndexes], y, cost = 10, cachesize=500,
scale=F, type="C-classification", kernel="linear" )

#compute the weight vector
w = t(svmModel$coefs)%*%svmModel$SV

#compute ranking criteria
rankingCriteria = w * w

#rank the features
ranking = sort(rankingCriteria, index.return = TRUE)$ix

#update feature ranked list
featureRankedList[rankedFeatureIndex] = survivingFeaturesIndexes[ranking[1]]
rankedFeatureIndex = rankedFeatureIndex - 1
#eliminate the feature with smallest ranking criterion
(survivingFeaturesIndexes = survivingFeaturesIndexes[-ranking[1]])

}

return (featureRankedList)
}
```

With this algorithm, now we can run the example of Breast Cancer Wisconsin dataset. We are using all the commands that we saw in the section 2.3:

```
x<-datos[,-1]
y<-datos[,1]
featureRankedList = svmrfeFeatureRanking(x,y)

featureRankedList
```

The command output is the set of features arranged in order of importance:

```
[1] 28 8 17 16 27 5 29 25 12 21 7 26 1 15 9 18 13 22 20 3 2 14 30 10 6

[26] 19 4 11 24 23
```

So, it is possible to select a specific number, less than the number of variables, of the most relevant features in order to train a better SVM model.

Embedded methods: The third approach, embedded methods, performs feature selection in the process of model building. For example, adds an extra term that penalizes the size of the selected feature subset to the standard cost function of SVM, and optimizes the new objective function to select features. These approaches are, however, limited to linear kernels.

Embedded methods have been proposed to reduce the classification of learning. They try to combine the advantages of both previous methods. The learning algorithm takes advantage of its own variable selection algorithm. So, it needs to know preliminary what a good selection is, which limits their exploitation.

Many generalizations of the original work of Boser, Guyon and Vapnik have been proposed. For instance, Bradley and Mangasarian, [5], describe the following general procedure. It is given to discriminate linearly separable data, like the problem (2.8):

$$\text{(3.1)} \qquad \begin{array}{ll} \min_{\beta_0,\beta} & \|\beta\|_p + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0+\beta^t x_i) \geq 1-\varepsilon_i,\ \forall\ i\in\mathcal{I} \\ & \beta\in\mathbb{R}^r, \beta_0\in\mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall\ i\in\mathcal{I} \end{array}$$

where $\|\beta\|_p = (\sum_{j=1}^n \beta_j^p)^{\frac{1}{p}}$ is the ℓ_p-norm of β. When $p=2$, this procedure is the same as the euclidean case above defined. The generalization capabilities of such linear models have been studied by many researchers who have shown that, roughly speaking, minimizing the ℓ_p-norm of β is good for generalization. We are going to study the case that is obtained when, in the previous problem, we apply the extreme case $p\to 0$, which in a slight abuse of terminology, the minimization of the zero-norm of β the latter being defined as:

$$\|\beta\|_0 = card\{\beta_i|\beta_i\neq 0\}$$

where *card* is set cardinality. Note that $\|.\|_0$ is not a norm because, unlike ℓ_p-norms with $p\geq 1$, the triangle inequality does not hold. Minimization of the zero-norm provides a natural way of directly addressing the feature selection and pattern classification objectives in a single optimization. However, this is achieved at the cost of having to solve a very difficult optimization problem which will not necessarily generalize well.

An approximation of the step function can be done, [28]:

$$\|\beta\|_0 = card\{\beta_i|\beta_i\neq 0\} \approx \textstyle\sum_i 1-e^{-\alpha|\beta_i|}$$

where α is a parameter that must be chosen. Bradley and Mangasarian, [4], suggest to set the value of α to 5, although it is also proposed (but in practice not attempted) to slowly increase the value of α in order to improve the aproximation. The authors showed that minimizing the above expression can be achieved by solving a sequence of linear programs of the following form:

$$\text{(3.2)} \qquad \begin{array}{ll} \min_{v} & \sum_{i\in\mathcal{I}}\alpha e^{-\alpha v_i^*(v_i-v_i^*)} + C\sum_{i\in\mathcal{I}}\varepsilon_i \\ \text{subject to:} & y_i(\beta_0+\beta^t x_i) \geq 1-\varepsilon_i,\ \forall\ i\in\mathcal{I} \\ & -v\leq\beta\leq v \\ & \beta\in\mathbb{R}^r, \beta_0\in\mathbb{R} \\ & \varepsilon_i \geq 0,\ \forall\ i\in\mathcal{I} \end{array}$$

where v^* is the solution v from the last iteration. Note that each of these iterations finds the steepest descent diection of the objective $\sum_{i\in\mathcal{I}}1-e^{-\alpha|\beta_i|}$ while keeping consistency with the constraints. It is proved that if α is sufficiently large then an optimal solution of problem (3.2) is also an optimal solution of the problem (3.1).

In Weston (2003), [28], is proposed a method called Approximation of the zero-norm Minimization (AROM):

```
1.Set $\underline{z} = (1, \cdots, 1)$

2.Solve:
```

$$
\begin{array}{ll}
\min_{\beta} & \sum_j^n |\beta_j| + C \sum_{i \in \mathcal{I}} \varepsilon_i \\
\text{subject to:} & y_i(\beta_0 + \beta^t (x_i * \underline{z})) \geq 1 - \varepsilon_i, \ \forall\, i \in \mathcal{I} \\
& \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\
& \varepsilon_i \geq 0, \ \forall\, i \in \mathcal{I}
\end{array} \tag{3.3}
$$

```
3.Let $\beta$ be the solution of the previous problem.  Set $\underline{z} \leftarrow \underline{z} * \beta$.

4.Go back to 2 until convergence.
```

It is simply a succession of linear programs combined with a multiplicative update. Sometimes, it can be adventageous to consider fast approximations of this algorithm. For instance, one could take a suboptimal descent direction by using the ℓ_2-norm instead of the ℓ_1-norm in step 2. The ℓ_2-norm has the advantage that it leads to a simple dual formulation which then becomes equivalent to training a SVM:

```
1.Set $\underline{z} = (1, \cdots, 1)$

2.Solve:
```

$$
\begin{array}{ll}
\max_{\lambda_i} & \sum_{i \in \mathcal{I}} \lambda_i - \frac{1}{2} \sum_{i,j \in \mathcal{I}} \lambda_i \lambda_j y_i y_j ((\underline{z} * x_i)(\underline{z} * x_j)) \\
\text{subject to:} & \sum_{i \in \mathcal{I}} \lambda_i y_i = 0, \ C \geq \lambda_i \geq 0, \ \forall\, i \in \mathcal{I}
\end{array} \tag{3.4}
$$

```
3.Let $\beta$ the solution of the previous problem.  Set $\underline{z} \leftarrow \underline{z} * \beta$.

4.Go back to 2 until convergence.
```

It is possible to extend the proposed algorithm to also trade off the training error with the number of feature selected, which is necessary in the linearly nonseparable case. For simplicity, let us again consider the case of two-class linear models. Introducing slack variables ε_i for each training point we can in general solve:

$$
\begin{array}{ll}
\min_{\beta} & \|\beta\|_p + C\|\varepsilon\|_q \\
\text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1 - \varepsilon_i, \ \forall\, i \in \mathcal{I} \\
& \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\
& \varepsilon_i \geq 0, \ \forall i \in \mathcal{I}
\end{array} \tag{3.5}
$$

Let us consider the case $p = 0$ and $q = 0$. Using this approach, it is possible to solve this problem by rewriting the training data $x_i \leftarrow (x_i, \frac{1}{\alpha}\delta_i)$ where $\delta_i \in \{0, 1\}^n$ and $(\delta_i)_j$=1 if $i = j$ and 0 otherwise.

Note that is also possible to derive a system to minimize the training error of SVMs, i.e. using $p = 2$ and $q = 0$. This is a kind of minimization that researchers have been interested in solving but have rarely in practice been able to solve, although Fung et al., [10], and Pérez-Cruz et al.,[23] , also propose solutions to this problem.

The method of approximately minimizing the zero-norm chooses a small number of features and can therefore be used as a feature selection algorithm. Features selection can be implemented usig this method in the following way:

$$\begin{array}{ll} \min_\beta & \|\beta\|_p + C\|\varepsilon\|_q \\ \text{subject to:} & y_i(\beta_0 + \beta^t x_i) \geq 1, \ \forall\, i \in \mathcal{I} \\ & \|\beta\|_0 \leq r - \varepsilon_i, \ \forall\, i \in \mathcal{I} \\ & \beta \in \mathbb{R}^r, \beta_0 \in \mathbb{R} \\ & \varepsilon_i \geq 0, \ \forall i \in \mathcal{I} \end{array} \tag{3.6}$$

for $p = \{1, 2\}$ and the desired number of features r. This method can be approximated by minimizing the zero-norm using the ℓ_2-AROM or ℓ_1-AROM methods, stopping the step-wise minimization when the constraint $\|\beta\|_0 \leq r$ is met. One can then re-train a p-norm classifier on the features which are the nonzero elements of β. In this way one is free to choose the parameter r which dictates how many features the classifier will use. This should thus be differentiated from a zero-norm classifier which would try to use the minimum number of features possible, which is not always the optimum choice in terms of generalization.

Bibliography

[1] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[2] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research*, 3:463–482, 2003.

[3] Kristin P Bennett and Olvi L Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, 1(1):23–34, 1992.

[4] Paul S Bradley and Olvi L Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.

[5] PS Bradley, OL Mangasarian, and JB Rosen. Parsimonious least norm approximation. *Computational Optimization and Applications*, 11(1):5–21, 1998.

[6] J Paul Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations Research*, 59(2):467–479, 2011.

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

[8] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Trading convexity for scalability. In *Proceedings of the 23rd international conference on Machine learning*, pages 201–208. ACM, 2006.

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

[10] Glenn M Fung, Olvi L Mangasarian, and Alexander J Smola. Minimal kernel classifiers. *The Journal of Machine Learning Research*, 3:303–321, 2003.

[11] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[12] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature Extraction*, pages 1–25. Springer, 2006.

[13] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[14] Ivana Hadzic and Vojislav Kecman. Support vector machines trained by linear programming: theory and application in image compression and data classification. In *Neural Network Applications in Electrical Engineering, 2000. NEUREL 2000. Proceedings of the 5th Seminar on*, pages 18–23. IEEE, 2000.

[15] 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.

[16] AJ Izenman. *Modern multivariate statistical techniques*, volume 1. Springer, 2008.

[17] Vojislav Kecman and Tiru Arthanari. Comparisons of qp and lp based learning from empirical data. In *Engineering of Intelligent Systems*, pages 326–332. Springer, 2001.

[18] Olvi L Mangasarian. Generalized support vector machines. *Advances in Neural Information Processing Systems*, pages 135–146, 1999.

[19] Olvi L Mangasarian and David R Musicant. Lagrangian support vector machines. *The Journal of Machine Learning Research*, 1:161–177, 2001.

[20] Carlotta Orsenigo and Carlo Vercellis. Multivariate classification trees based on minimum features discrete support vector machines. *IMA Journal of Management Mathematics*, 14(3):221–234, 2003.

[21] Carlotta Orsenigo and Carlo Vercellis. Evaluating membership functions for fuzzy discrete svm. In *Applications of Fuzzy Sets Theory*, pages 187–194. Springer, 2007.

[22] Carlotta Orsenigo and Carlo Vercellis. Softening the margin in discrete svm. In *Advances in Data Mining. Theoretical Aspects and Applications*, pages 49–62. Springer, 2007.

[23] Fernando Pérez-Cruz, Angel Navia-Vázquez, Aníbal R Figueiras-Vidal, and Antonio Artés-Rodríguez. Empirical risk minimization for support vector classifiers. *Neural Networks, IEEE Transactions on*, 14(2):296–303, 2003.

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

[25] Xiaotong Shen, George C Tseng, Xuegong Zhang, and Wing Hung Wong. On ψ-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

[26] Berk Ustun, Stefano Traca, and Cynthia Rudin. Supersparse linear integer models for predictive scoring systems. *arXiv preprint arXiv:1306.5860*, 2013.

[27] Zhuang Wang and Slobodan Vucetic. Fast online training of ramp loss support vector machines. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 569–577. IEEE, 2009.

[28] Jason Weston, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. Use of the zero norm with linear models and kernel methods. *The Journal of Machine Learning Research*, 3:1439–1461, 2003.

[29] Weida Zhou, Li Zhang, and Licheng Jiao. Linear programming support vector machines. *Pattern recognition*, 35(12):2927–2936, 2002.